2e édition corrigée et augmentée

Le fil d'ariane

tome II

microprocesseurs et conception des petits ordinateurs

par Jean-Pierre Bouhot, Georges Cottin
et Jean Tricot

Préface de Jean-Claude Pelissolo
Directeur des Industries Electroniques et Informatiques
au Ministère de l'Industrie

Editions d'Informatique

Un Fil d'Ariane, tome II
Microprocesseurs et Conception des petits ordinateurs

LES OUVRAGES DES EDITIONS D'INFORMATIQUE

Un Fil d'Ariane (tome I) : l'ordinateur à la portée de tous
par J.-P. Bouhot et M. Péju
Le best-seller de l'initiation à l'informatique. Troisième édition avec un nouveau chapitre. Adopté par Peugeot, EDF, la Sécurité Sociale, etc.

La micro-informatique : introduction aux systèmes répartis
par B. Lussato, J.-P. Bouhot et B. France-Lanord
L'invasion des petits ordinateurs face aux gros : l'ouvrage de référence.

L'ordinateur de la banque
par A. Ruff et Ph. Bellan
L'informatique dans le système bancaire : heurs et malheurs...

Le Cobol structuré
par M.-T. Bertini et Y. Tallineau
La méthode de programmation qui révolutionne la programmation Cobol.

De la saisie à la source à l'informatique répartie
par P. Samson
Les solutions aux problèmes de la saisie des données.

Objectifs et conception d'une base de données
par J. Thuly et A. Saunier
Approche méthodologique et application pratique.

La programmation structurée
par un groupe d'experts
Les auteurs les plus compétents et les plus prestigieux font tout connaître sur la programmation structurée.

VIENT DE PARAITRE

Rapports Spéciaux
par R.-C. Canning
Chacun de ces rapports s'attache à un domaine informatique du plus haut intérêt par une analyse qui concerne tous les niveaux de l'entreprise, du plus bas au plus élevé.

ISBN 2-901001-09-2

Copyright
Editions d'Informatique

Jean-Pierre BOUHOT - Georges COTTIN
Jean TRICOT

Un Fil d'Ariane, tome II
Microprocesseurs et Conception des petits ordinateurs
(2e édition)

Préface de Jean-Claude PELISSOLO
Directeur des Industries Electroniques et de l'Informatique
au Ministère de l'Industrie

EDITIONS D'INFORMATIQUE
99, bd Jean-Jaurès - 92100 BOULOGNE
Tél. : (1) 604.07.08

PREFACE

Il est extrêmement difficile d'expliquer simplement et clairement ce qu'on connaît bien.

Cet exercice est encore plus périlleux lorsqu'il s'applique à un objet aussi énigmatique pour la profane que le microprocesseur.

Jean-Pierre BOUHOT, Georges COTTIN et Jean TRICOT s'en tirent avec élégance et maestria, en virtuoses de la littérature technique, et se paient le luxe de jalonner d'humour le chemin de la découverte.

Car c'est bien de découverte qu'il s'agit. Et j'imagine le bateleur de l'an 2050 : "Ecoutez Mesdames et Messieurs, l'histoire de l'invention la plus fabuleuse du 20e siècle, qui a révolutionné la vie de nos grands parents en mettant l'intelligence en bouteille !".

En bouteille oui, et même en capsule, car c'est sur quelques millimètres carrés de silicium que, grâce à la technique des circuits intégrés, on sait loger, pour quelques dizaines de francs, une puissance de calcul et de mémorisation supérieure à celle des grandes machines à 1 million de dollars d'il y a 20 ans.

Si on avait voulu, en 1960, reproduire l'ensemble des connexions logiques contenues dans le cerveau humain, il y aurait fallu une surface égale à celle de Paris. Il y a dix ans, la surface de Notre-Dame aurait suffi. Aujourd'hui, celle d'une chambre à coucher. Demain, avant cinq ans, le volume... d'un cerveau humain.

Ces microprocesseurs travaillent déjà au rythme de plusieurs millions d'opérations par seconde, et le produit rapidité $\times$ densité croît régulièrement, sans qu'on puisse encore déterminer l'asymptote.

Où nous conduit ce vertigineux exploit technique ? Que rend-il possible ? Avant tout, la dissémination de l'intelligence dans tous les produits, ou presque, créés par l'homme.

C'est le mythe de Pinocchio devenu... presque... réalité. Et le souffle de vie, qui d'une marionnette inanimée fait un pantin intelligent et adroit, c'est ... la programmation. Le microprocesseur, lorsqu'il arrive sur la table de son utilisateur, est un petit morceau de plastique inerte, en tout point semblable à des millions d'autres. Son extraordinaire puissance logique sera révélée par le programme, élaboré sur mesure, et avalé par sa mémoire aussi simplement que la bande perforée par l'orgue de barbarie.

Ainsi personnalisé, le microprocesseur pourra à notre guise piloter une moissonneuse-batteuse, transcrire pour un aveugle l'image en sons, reconnaître les empreintes digitales, commander l'allumage du moteur de votre automobile, agir sur ses freins pour empêcher les roues de déraper, réaliser des variations sur un thème de Bach ou de Varèse, et répertorier les mille autres exemples d'applications des microprocesseurs !

Cet instrument prodigieux sera demain si répandu qu'il en deviendra banal : un ingénieur, électronicien ou non, ne pourra pas plus ignorer son fonctionnement qu'un électricien le principe du va-et-vient ou un mécanicien celui du différentiel.

C'est donc à l'Université, aux Ecoles d'Ingénieurs, sans perdre une année, qu'il revient de former à l'utilisation des microprocesseurs des générations entières d'étudiants.

Quant aux adultes, ils vont devoir se ''recycler''. Mot horrible qui évoque les affres et la transpiration du candidat à la reconversion technique, plein de complexes devant le jeune diplômé goguenard censé lui réapprendre son métier.

En fait, cet ouvrage montre, et ce n'est pas son moindre mérite, que ces nouvelles techniques peuvent être apprivoisées sans effort intellectuel épuisant. Par des routes agréables, à pente douce, et dont les lacets vous feront découvrir des paysages inattendus, il vous conduira insensiblement vers les sommets de la technologie, d'où vous apercevrez, dernier chapitre du livre, ... l'horizon de la micro-électronique : l'imagination créatrice de l'homme.

Permettez-moi d'espérer que, redescendu dans la plaine, vous aurez envie d'utiliser, vous aussi, l'instrument dont ce livre vous aura révélé la puissance.

Jean-Claude PELISSOLO
Directeur des Industries Electroniques
et de l'Informatique
au Ministère de l'Industrie

AVANT-PROPOS

Le texte que vous allez lire présente une caractéristique unique: dès la première page, vous allez partir avec nous *à la merveilleuse découverte des microprocesseurs , exactement celle que nous avons vécue. Il s'agit en effet d'une chronique, au sens premier du terme. Çà et là vous trouverez des références permettant de dater le récit... qui s'étend de Décembre 1974 à Juillet 1979 (1): pire que Ponson du Terrail et à peu près aussi mouvementé...*

Remercions ici Jean-Claude PELISSOLO de sa sympathique préface, qui dit fort bien à quel point le microprocesseur, sujet principal de notre travail, est une invention fabuleuse.

Nous n'avons apporté qu'un minimum de corrections au texte, afin de ne pas dénaturer son actualité dans le temps, à chaque phase d'évolution des microprocesseurs. Ceci nous a semblé important, pour permettre au lecteur de bien se plonger dans cette progression qui continue, *en saisissant exactement son rythme, d'une constance superbe. Comme dans le précédent* Fil d'Ariane, *le lecteur est mené pas à pas, de façon tout à fait intelligible, depuis un niveau de départ élémentaire jusqu'à une compréhension poussée de sujets réputés difficiles.*

De chapitre en chapitre, après les découvertes des semiconducteurs, de la logique et des microprocesseurs de toute nature, on arrive à la conception par

---

(1) Il a paru en feuilleton dans l'Informatique Nouvelle (dans des numéros maintenant épuisés) jusqu'au chapitre 8-2 inclus. La suite est inédite.

étapes d'un petit ordinateur. Ce n'est pas un cas d'école, la machine existe réellement : c'est même le premier ordinateur professionnel de gestion, réalisé autour d'un microprocesseur, qui, dans le monde, ait été mis à la disposition des utilisateurs sous forme de kit. Aujourd'hui, ce système n'existe plus que monté, vu son extension. Mais nous avons conservé les chapitres relatifs au kit (tout en élaguant les explications extrêmement détaillées de montage dont nous avons supprimé les trois quarts), car d'une part ce sont eux qui expliquent le mieux la conception, et d'autre part tous ceux qui seront appelés à concevoir, étudier, manipuler, monter ou dépanner des micro-ordinateurs trouveront dans les parties du montage que nous avons laissées une foule de conseils et de remarques pratiques, d'exemples d'implantation de cartes, de dispositions de fond de paniers, etc.

Nous avons reproduit intégralement les détails de programmation du chapitre 7-7, à titre d'exemple rarement donné de la réalisation d'un driver de périphérique : on verra la finesse de programmation nécéssaire, très instructive. Cette expérience est directement transposable à d'autres périphériques.

De même, tous les textes concernant les entrées/sorties sont in extenso, *permettant une bonne approche de ces usines à gaz que sont les chips périphériques tels que le 8255, ainsi que du mode toujours un peu mystérieux des télécommunications et des transmissions de données.*

A partir du chapitre 8, une excursion nous emmène dans le domaine des gros ordinateurs pour explorer les problèmes particuliers que l'avènement de la micro-électronique a posés pour leur conception, ainsi que les solutions nouvelles qui émergent, parfois diamétralement opposées comme on le verra chez IBM et chez Univac. Avec en plus un regard sur le lointain avenir...

Le chapitre 9, lui, est entièrement dévolu aux nouveaux microprocesseurs, qui permettent encore un autre bond en avant. Ne déflorons pas le sujet, il est trop extraordinaire.

Enfin, en annexe, un texte d'une nature assez rare pourra vous permettre les premiers pas dans l'écriture d'un logiciel système pour microprocesseur. Il n'y a rien de tel que de forger aussi *ses propres outils logiciel : c'est la seule manière de maîtriser vraiment les microprocesseurs.*

Et maintenant, nous vous souhaitons autant de plaisir à lire ces pages que nous en avons eu nous-mêmes à les écrire pour vous !

CHAPITRE 1
LES DEUX CULTURES

Super Zap dérivait doucement à ras du plafond dans une zone de pénombre. Une brève pression sur le vernier du dégraviteur l'immobilisa, juste au-dessus du prototype de l'Ordinateur qui s'étalait sans défense, son anatomie électronique à nu, tous capots ôtés.

Le moment ou jamais.

Il déclencha l'obturateur de son stéréo-neutrinographe, et les deux images — la visible et l'invisible — se fondirent dans la microplaque de silicium feuilleté. C'était fait.

Le rythmeur temporel réglé à un kilohertz, Super Zap passa en mode conditionnel, alternativement en avance et en retard d'une microseconde sur le temps présent, et s'enfonça dans le brouillard de l'épaisse coupole de béton...

... Une heure après, le triple développement ionodiffusant terminé, la micro-machine était prête. Extraordinaire labyrinthe microscopique, reproduisant dans ses moindres détails l'Ordinateur adverse dont elle pouvait désormais assumer toutes les fonctions, bien qu'un milliard de fois plus petite, aussitôt animée par les électrons parcourant ses nerfs elle commença à parler à son maître, et illumina les écrans...

Science-fiction ? Oui... et non : car cette micromachine — ou du moins sa jeune sœur — nous sommes en train de l'examiner sous un projecteur. Au grossissement 32 fois, une structure prodigieusement complexe apparaît, évoquant une vue aérienne des allées et des orangeraies d'un immense jardin à la française, les ruines de Macchu-Pichu ou les voies de la gare de triage de Trappes, avec des milliers de wagons en attente.

Vingt ans seulement ont passé, et cette microplaquette de silicium de quelques milligrammes remplit davantage de fonctions, bien plus rapidement, que le processeur d'un ordinateur de première génération pesant une trentaine de tonnes. Même en présence d'un tel microprocesseur, nous avons peine à croire nos yeux et notre raison.

C'est là un des plus merveilleux produits de la technologie de pointe, riche d'extraordinaires promesses : notre but va être de l'étudier, pour pouvoir le comprendre et enfin l'utiliser.

1 - 1. LE MONDE MERVEILLEUX DU MICROPROCESSEUR

En trois ans, déjà la deuxième génération de microprocesseurs... Mais quittons un instant le moyen pour considérer les fins. Que peut signifier la possibilité de réalisation qui s'offre, dès aujourd'hui, d'un micro-ordinateur complet coûtant à fabriquer quelques milliers de francs avec ses périphériques ? Les microprocesseurs de seconde génération, grâce à leur vitesse très élevée, et au faible coût des mémoires à circuits intégrés, ouvrent la possibilité d'un éventail si vaste d'applications (économiquement impensables pour leurs chers grands frères) qu'on en est presque saisi de vertige.

Essayons de passer en revue quelques classes d'utilisateurs potentiels d'une micro-machine comportant dans sa version de base une visualisation à diodes électro-luminescentes (comme celles des calculatrices de bureau), un clavier et une mini-cassette magnétique de stockage de programmes (naturellement, une série d'extensions y seraient prévues : lecteur de cartes ultra-simple, imprimantes numériques, alpha-numériques, disques souples, d'autres périphériques étonnants et surtout une connexion directe sur récepteur de télévision du commerce par la prise d'antenne, avec photo-style optionnel pour désignation d'une zone de la visualisation) :

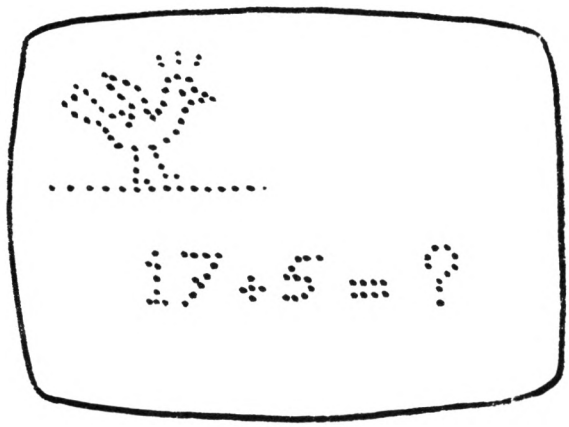
— **Services informatiques.** A côté des grosses machines, le micro-ordinateur pourra jouer un précieux rôle d'auxiliaire pour les programmeurs, les programmeurs système, le chef programmeur, le responsable du service et même le chef d'exploitation :
  - aide au dépouillement des listings de programmes ou de dumps en hexadécimal, octal, conversions multiples entre des systèmes et le décimal, le binaire, etc.,
  - calcul des exemples numériques avant essais de programmes,
  - formation à la programmation des programmeurs débutants, ou des utilisateurs dans le cadre d'une action plus générale dans les services, au moyen de la simulation d'une machine virtuelle décimale simple dotée d'un petit nombre d'instructions (une dizaine), comme le Boufalac par exemple (1),
  - formation à la programmation fine, ordres logiques booléens, arithmétique binaire et BCD, tables de vérité et de conditions, etc.,
  - tests d'aptitude de programmation, avec affichage des données du test, mesure du temps de réponse, pondération et cumul des résultats,
  - tests de connaissances en programmation avant embauche d'un nouveau programmeur, à qui l'on remet la feuille d'instructions d'une machine virtuelle ainsi que les spécifications d'un programme à réaliser ; mesure du temps d'écriture et de mise au point, ainsi que du temps de déroulement,
  - gestion des programmes de bibliothèque,
  - analyseurs syntaxiques de langages permettant la programmation on-line, un premier niveau de vérification, le contrôle immédiat du respect du sous-ensemble normalisé du langage (Cobol C), l'élimination des erreurs de perforation, tout ceci avec un mode de travail plus agréable pour le programmeur, et une économie du temps de la grosse machine,
  - les mille et une autres idées que vous n'allez pas manquer de nous envoyer : nous les attendons avec intérêt, écrivez-nous !

— **Bureaux d'étude et de calcul.** Il est difficile ici d'être spécifique tant les domaines existants sont variés. Le service informatique pourra réaliser une ou des machines pour le bureau d'étude, ou ce dernier la monter lui-même. Avec les fonctions arithmétiques, logarithmiques, trigonométriques et de conversion de systèmes de coordonnées, alliées à la possibilité de stocker des programmes conséquents, les services rendus par la micro-machine seront innombrables, des calculs de cotes et d'angles aux calculs d'engrenages, des dessins de moules aux calculs de performances, de la résistance des matériaux aux calculs de circuits électriques ou électroniques en passant par ceux de chauffage, les dépouillements d'essais, etc.

— **Etablissements d'enseignement.** Le domaine laisse ici rêveur tant les possibilités sont grandes, de la maternelle aux classes les plus élevées, tout spécialement avec la connexion télévision permettant une interaction directe, enrichie de la possibilité de reproduction d'enregistrements sonores, vocaux, musicaux ou autres par la cassette, sous contrôle du programme :
  - corrélation des mots (affichés TV, lus sur cartes) et de leur énoncé vocal, idem pour sons, objets TV,
  - orthographe en plus, par frappe au clavier,
  - tests de lecture,
  - pour les plus jeunes, reconnaissance des mouvements droite et gauche, haut et bas, et manipulation par boutons fléchés, promenade de la souris dans le labyrinthe, etc.
  - entraînement à la mémorisation par le jeu de Kim et dérivés,
  - exercices élémentaires d'arithmétique, pendant que l'oiseau s'envole pour s'échapper. Si l'on va assez vite, il redescend et se repose. Cumul des temps, notation,
  - exercices de logique de 7 à 77 ans, tels qu'une visualisation du Wff'n proof avec la notation

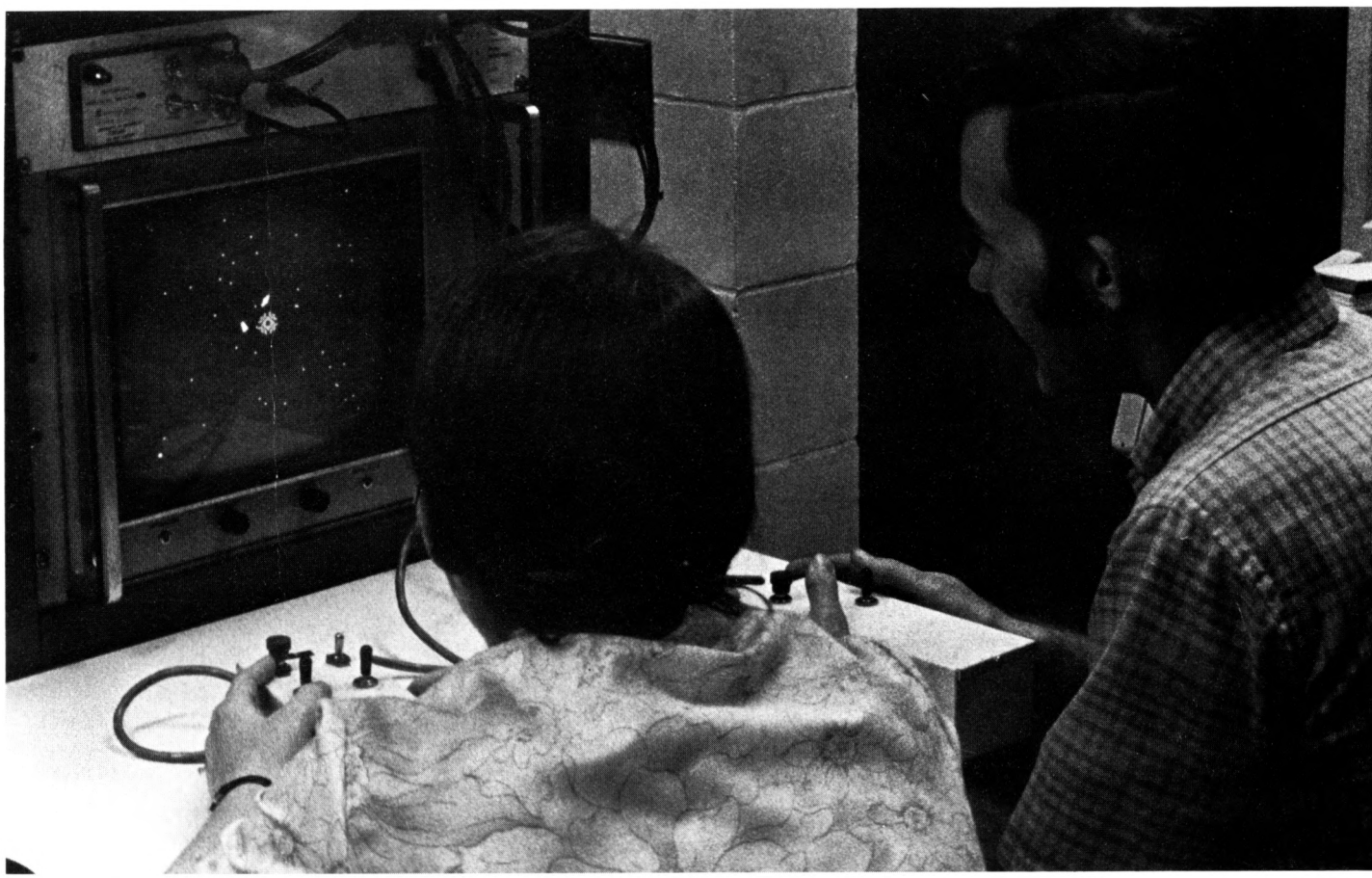
(1) Voir : *Un Fil d'Ariane : l'ordinateur à la portée de tous*, par J.-P. Bouhot, Editions d'Informatique (4e édition avec index et chapitre additionnel, 1979).

polonaise de Lukasiewicz et le test syntaxique des expressions,
- algèbre de Boole et tables de vérité,
- simulateur logique (version programmée des pupitres à fiches actuellement en utilisation),



- innombrables tests de connaissances avec commentaires parlés choisis par saut logique sur la cassette selon réponse,
- utilisation de l'écran pour des jeux de classe par équipe,
- exercices de conversion de nombres d'une base dans une autre (mêmes sous-programmes que pour le service informatique !),
- visualisation de fonctions, d'approximations,
- calculs de séries avec visualisation de convergence rythmée,
- principes fondamentaux de l'informatique par la machine de Turing visualisée dans ses diverses versions (2),
- apprentissage élémentaire de la programmation sur machine virtuelle simple (Boufalac, voir service informatique),
- apprentissage de la programmation niveau machine de bureau d'étude avec fonctions variées,
- apprentissage avancé de la programmation, soit par machine virtuelle adéquate, soit directement avec les instructions du microprocesseur (possibilité de création de software performant),

(2) «La Machine de Turing additionne en comptant sur ses doigts». *L'Informatique nouvelle* n° 35 décembre 1972 (voir note suivante).



C'est un jeu fascinant, sans doute le premier de l'ère de l'espace. Son inconvénient majeur est de nécessiter, dans sa version minimale, quelque vingt mille dollars d'ordinateur et d'équipement périphérique. En fait une version interessante reclame environ deux cent cinquante mille dollars (de l'ordre de un million de francs actuels). Bien peu d'individus en disposent... (Albert W. Kuhfeld, Analog, *Juillet 1971, p. 67 et seq.)*

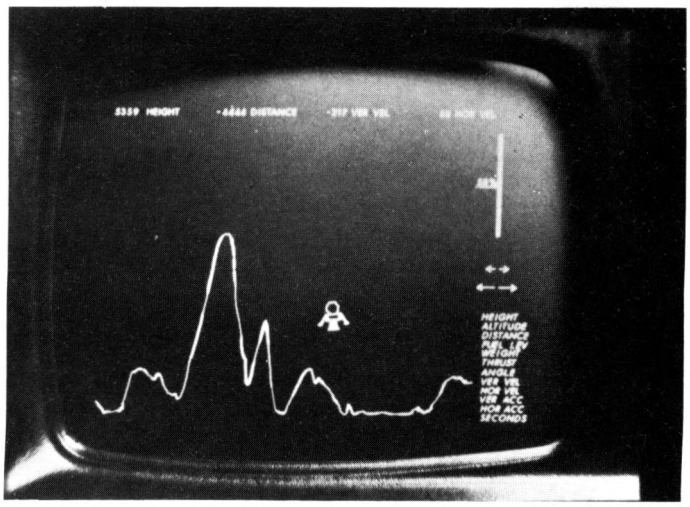
- les mille et une autres idées que vous n'allez pas manquer de nous envoyer : nous les attendons avec intérêt, écrivez-nous !

— **Services d'entreprises.** Là encore la diversité est telle qu'on ne peut que citer au hasard : calculs économiques, calculs statistiques, affichage de graphiques à barres photographiables en Polaroïd, coûts de revient, études de rentabilité, de programmes d'investissements, secrétariat automatique avec édition/correction de textes, etc. De nombreux programmes d'aide à la décision qui s'écrivent actuellement sur des consoles de time-sharing seront avantageusement développés et exploités sur le micro-ordinateur : l'expérience pratique montre d'ailleurs que les utilisateurs de temps-partagé n'emploient guère plus de 20 % de programmes de bibliothèque et préfèrent écrire les leurs. Seuls ceux d'un gros volume ou faisant appel à des fichiers resteront justiciables du temps-partagé, ce qui est la voie d'avenir normale des réseaux de traitement.

— **PME/PMI.** Ici, le micro-ordinateur sera le successeur — ô combien plus pratique et puissant — de l'ordinateur de bureau. Transactionnel par nature même, avec dialogue à l'écran, il se révèlera à moyen terme comme l'outil idéal des PME.

— **Jeux (éducatifs et autres).** Décidément, nous allons commencer à manquer de superlatifs. Un coup d'œil sur la chronique «Jeux de la machine» (3) suffit à nous convaincre du nombre et de la variété des applications possibles. Notons que nombreux sont les jeux récréationnels qui ont cependant un caractère éducatif marqué, comme on le verra au hasard de la liste très partielle ci-après (c'était déjà le cas des jeux de logique cités) : ils ont donc également leur place dans la formation et l'enseignement (et peut-être ailleurs !). Cette seule catégorie de jeux comporte plusieurs centaines de titres... Notons d'autre part que vous pourrez jouer avec Nanostar comme opposant (ça y est, nous avons lâché son nom) tel Dave et HAL 9000 dans 2001 (la visu sera un peu moins glorieuse et les échecs ne sont pas envisagés pour l'instant) ou à plusieurs joueurs selon nature du jeu et/ou option choisie. Ayant déjà programmé bon nombre de ces jeux, nous savons que les encombrements des programmes sont réalistes pour la machine :

- bataille spatiale (4) avec orbites, poursuite, consommation de carburant, tirs d'engins,
- dés électroniques, avec générateur de nombres aléatoires,
- Tic-Tac-Toe, avec changement possible de règles, version heuristique (5) et machine à apprentissage,
- jeux de pions de toutes sortes, dont par exemple l'Hexapion, inventé par Martin Gardner,
- ver hexamétrique de Beeler, avec tracé progressif visualisé (cf. laboratoire d'intelligence artificielle du MIT, notre correspondance avec Marvin Minsky),
- puzzles à pièces glissantes (innombrables : puzzles orientaux, taquin, âne rouge, polyminoes de toutes sortes, avec comptage des coups),
- parcours et tracés de labyrinthes, changeants, invisibles, à un ou plusieurs joueurs,
- circuits, vitesse variable, accélération/freinage/inertie/sol plus ou moins glissant,
- football électronique,
- jeux de Nim et de Marienbad (6),
- pétanque et bowling (effets sonores, irrégularité aléatoire du sol),
- les images animées simples pour les enfants,
- jeu de la vie, de Conway, où les structures se développent, meurent, se déplacent (passionnant, a consommé des dizaines de milliers d'heures d'ordinateurs d'un bout à l'autre des USA),
- jeux de simulation, *innombrables*, modelant l'univers physique, économique, etc., de la bataille spatiale et d'autres jeux déjà cités à des jeux d'entreprise (le Monopoly en est un), de la lutte contre la flotte des pirates à l'atterrissage (sic) lunaire où l'on pilote le LEM de la High Gate à la Low Gate jusqu'au touch-down final.



Suspense : le LEM arrivera-t-il à se poser en douceur avant épuisement des propergols ? Tout dépend de votre manœuvre...

(3) Mensuelle, par Jean Tricot, dans *l'Informatique nouvelle*, Editions d'Informatique.

(4) op. cit. «La Bataille spatiale», *L'Informatique nouvelle* n° 40 et 41, mai et juin 1973.

(5) op. cit. «Prolégomènes à l'étude du Tic-Tac-Toe», «MENACE sur le Tic-Tac-Toe», *L'Informatique nouvelle* n° 51, 52 et 53, mai, juin et juillet/août 1974.

(6) op. cit. «Le jeu de Nim : De Marienbad au système binaire», *L'Informatique nouvelle* n° 56 novembre 1974.

- les mille et une idées que vous n'allez pas manquer de nous envoyer : nous les attendons avec intérêt, écrivez-nous !

Nous arrêtons volontairement ici la liste des jeux qui paraît autrement ne pas avoir d'extrémité en vue à distance finie, si l'on tient compte que l'on peut combiner les possibilités sonores (sons enregistrés sur la cassette ou sons synthétiques créés par des boucles de programme tournant à vitesse variable) et visuelles (statiques, mobiles, clignotantes) à une logique de programme déjà étoffée. A nous le calculateur d'annonces au bridge, le professeur de morse de plus en plus virtuose, le découvreur du mot caché, la machine à sous électronique à cloches, citrons et cerises, le meneur de jeu aux questions à tiroirs et au métronome infatigable, arrêtons là, le plus sceptique est déjà k.o. sur ce plan mais se pose déjà, probablement, quelques autres questions.

1 - 2. D'OU VENONS-NOUS, OU ALLONS-NOUS ?

Tout ceci est-il une affaire sérieuse ou une pure galéjade ? La technologie a-t-elle *réellement* évolué au point qu'un champ aussi vaste d'applications, jusqu'à il y a peu réservées aux grosses machines coûteuses et par conséquent à peine diffusées (l'article cité d'Arnold Kuhfeld sur la bataille spatiale ne s'intitulait-il pas «*Un échiquier de 250.000 dollars*»), s'ouvre à notre microprocesseur ? Essayons d'aligner quelques chiffres, en jetant spécialement un coup d'œil au passage sur ce qui s'est produit dans le domaine de l'informatique de gestion.

On ne peut manquer d'être extrêmement impressionné par la rapidité d'évolution actuelle du hardware. Il a en quelque sorte retrouvé un second souffle. Pendant une ou deux décennies nous avons eu l'impression que les progrès électroniques étaient assez lents. Nous sommes passés par la série des générations bien connues :

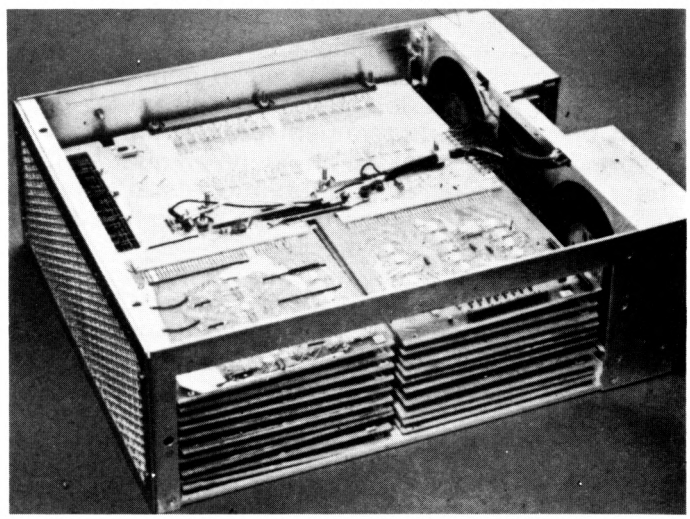
— La première génération avec des machines à tubes à vide, les lampes radio, dont la mémoire centrale était constituée par des tambours magnétiques. Vitesse pratique : 300 instructions : par seconde, prix de vente en Francs constants deux à trois millions actuels. Ceci a duré jusque vers les années 1958-1959.

— Puis s'est produit un saut brutal avec l'apparition des machines de la seconde génération. Le 1 401 est la machine la plus représentative de cette époque, ordinateur de puissance moyenne brusquement dix à quinze fois plus rapide que la génération précédente, à prix à peine supérieur. Ce saut d'ailleurs n'était pas vraiment dû à la partie logique et électronique des machines. Les tubes ont été remplacés par des transistors, ce qui a amené une amélioration de fiabilité considérable (1958 : la série General Electric 100 pour la Bank of America, date clé). Les matériels étaient plus compacts, certes, mais c'est surtout le changement technologique de mémoire centrale, le passage du tambour magnétique tournant à la mémoire statique à tores de ferrite qui a provoqué cette brusque accélération de la vitesse de traitement des matériels. Dans les machines à tambour, on peut en fait dire que la logique de l'unité centrale passait presque 90 % de son temps à attendre quelque chose : à attendre l'instruction suivante ou à attendre une donnée à traiter.

Et là les conséquences sur notre profession ont été assez brutales. Les machines de la première génération pour les travaux de gestion n'étaient que peu compétitives avec le matériel déjà en place, c'est-à-dire le matériel à cartes perforées traditionnel et les calculatrices électroniques à diodes du type du Gamma 3B par exemple, ou les machines IBM à tubes de la même époque, les 604 et autres.

Il s'en est suivi une véritable explosion de l'informatique. C'est seulement à partir des années 1960 que les ordinateurs ont commencé de tout envahir et que la mécanographie a été en brutale régression.

— Ensuite, pendant une dizaine d'années, les progrès ont été très mesurés. On est resté à la technologie du transistor, on est resté à la mémoire à ferrite. Et c'est la raison pour laquelle il n'y a pas eu d'autre brusque saut. Le temps d'accès à un mot en mémoire centrale est passé de 15 à 20 microsecondes à 10 microsecondes, puis à 5 microsecondes et lentement on s'est acheminé vers ce qui est à peu près un plancher économique compris entre 1/2 et 1 microseconde. La troisième génération a ainsi atteint 15.000 instructions par seconde vers 1966, puis (toujours dans la même classe de machines «moyenne basse» que nous prenons comme référence depuis le début) 35.000



La première mémoire à tores de ferrite n'utilisant que des circuits intégrés pour sa partie électronique (16.384 mots, cycle 500 ns, Honewell Computer Control Division). Dix fois moins d'interconnexions, d'où une fiabilité très accrue.

instructions par seconde vers 1972. Donc plus beaucoup de progrès à attendre de ce côté-là. Et tout le monde attendait les circuits intégrés. On lisait les articles aéro-spatiaux en particulier (c'est toujours là qu'il faut regarder) la technique aéro-spatiale précède de pas mal d'années la technique civile. Bref, on savait que ces circuits existaient, mais il était difficile de mesurer leur maturité.

— L'impact est arrivé à la fois de manière directe sur le plan de la logique des machines, et détournée sur le plan des mémoires.

Sur le plan de la logique, il s'est manifesté non pas par une brusque accélération de la rapidité de traitement dans les machines, mais par une baisse absolument spectaculaire du coût des circuits logiques pour l'unité centrale. Ceci a été l'effet numéro Un.

Le second effet s'est manifesté au niveau des mémoires par la part la plus en plus grande que la technologie MOS a commencé à se tailler dans le domaine des mémoires centales. Petit à petit les tores ont été grignotés, mais ce qui est particulièrement significatif, c'est que l'adoption de cette technologie MOS a permis d'un seul coup de diminuer le facteur négatif qui apparaissait dans le cas de fractionnement des mémoires. En effet, une mémoire à ferrite est composée certes de plusieurs centaines de milliers de tores, mais surtout d'une électronique associée qui est très importante. D'où la raison pour laquelle le coût du bit stocké était très inférieur pour les mémoires à ferrite de grande taille. On n'avait qu'une seule électronique de commande et on l'amortissait sur un très grand nombre de bits stockés en mémoire. Puis l'utilisation des circuits intégrés au lieu de transistors pour cette électronique de commande a d'abord permis d'en abaisser considérablement le coût.

Avec enfin l'apparition des circuits intégrés MOS à grande échelle remplaçant cette fois les tores eux-mêmes, nous avons vu le phénomène s'accentuer. Le coût des mémoires devient maintenant en pratique directement proportionnel au nombre de bits stockés. Si on coupe en quatre ou cinq une mémoire à circuits intégrés d'assez grande taille, on a des mémoires individuelles qui sont sensiblement quatre ou cinq fois moins chères que la grande, et non pas seulement par exemple la moitié du coût de la grande. Combinons tout ceci à l'effet numéro Un : il s'en est suivi qu'a pu d'abord être possible l'apparition sur le marché du mini-ordinateur à tores de ferrites commandés par circuits intégrés, puis enfin du micro-ordinateur, comportant une unité de traitement complexe avec sa mémoire centrale entièrement en circuits intégrés MOS, à un coût et un rapport coût/performances sans commune mesure avec ceux des ordinateurs des lignes traditionnelles.

Pourquoi le fait est-il passé inaperçu dans sa première phase ? Ceci est dû tout d'abord au marché choisi par les fabricants, qui était un marché de laboratoire, scientifique. Le marché de gestion a été ignoré. De telle sorte que les informaticiens de gestion n'ont pas pu faire de comparaison directe entre les rapports performances sur coût des mini-ordinateurs et des ordinateurs traditionnels.

Mais le tableau est en train de changer soudainement. Les constructeurs de mini-ordinateurs se sont rendus compte que le marché de la gestion était très important, et tous sont maintenant en train de préparer des softwares de gestion. C'est peut-être un bien grand mot, disons des logiciels de base, avec le minimum qu'on peut demander à une machine sur le plan système d'exploitation, plus un certain nombre de modules standards de traitement.

Un autre facteur négatif, de nature psychologique, est le nom qui a été donné à ces machines. Nous pensons qu'il a été désastreux d'appeler ces engins des mini-ordinateurs. Mini a une résonance péjorative, et dans l'esprit de tout un chacun a été associé aux performances de ces machines, alors qu'en fait c'est simplement à leur taille

(on aurait dû dire des «compacts») qu'il aurait fallu associer le terme. Ce sont des mini machines sur le plan de la taille, mais elles sont de taille au moins moyenne, sinon grande, en ce qui concerne les performances.

Nous en arrivons donc à l'heure actuelle où nous avons eu tout d'abord des microprocesseurs sur un seul circuit intégré, de vitesse supérieure à 10.000 instructions (opérant sur un ou deux caractères) par seconde, avec des mémoires entièrement électroniques, une baisse continue du coût des circuits et du coût des mémoires, et pour l'avenir un grand point d'interrogation. Que nous réserve-t-il ?

Avant de conclure, jetons un coup d'œil sur ce qui s'est passé en informatique de gestion : pendant des années nous avons sans arrêt concentré des traitements sur nos matériels. Il y avait une bonne raison à cela : tous les cinq ou six ans les constructeurs d'ordinateurs nous présentaient de nouvelles machines, en nous disant que ces nouvelles machines étaient beaucoup plus intéressantes car permettant un coût d'exécution de l'instruction plus faible. Et c'était vrai. Mais en même temps elles étaient plus grosses. On était donc obligé d'adopter des machines plus grosses pour avoir des coûts de traitement plus faibles. Et ce mouvement n'a pas arrêté pendant une quinzaine d'années. Un seuil a été franchi à un moment, à savoir que l'augmentation continue de la vitesse de ces machines faisait que nous avons dépassé le point à partir duquel toute augmentation ultérieure de vitesse de traitement interne ne se reflétait plus par des décroissances de coût unitaire de traitement, freiné comme nous l'étions par les vitesses des périphériques (ceci en applications de gestion).

Nous avons procédé ailleurs à une démonstration de ce fait, en nous cantonnant au domaine des tâches de gestion d'importance moyenne (excluant les systèmes géants tels que les systèmes de réservation de places ou ceux des organisations qui par nécessité doivent être centralisées et traitent des quantités considérables de gros travaux). Pour lutter contre ce fait, nous avons utilisé la multiprogrammation qui a fini par produire un software étouffant.

Vu l'état actuel de la technologie, nous avons pu dire que la loi de Grosch (la loi d'économie d'échelle) est morte, et avons proposé de la remplacer par une autre loi qui la reprend en sens exactement opposé. Loi de Bouhot : «*Le coût le plus bas d'un travail de gestion moyen est obtenu avec le plus petit ordinateur (monoprogrammé) l'exécutant dans les délais nécessaires*».

L'avenir nous permettra de trancher (7), mais nous pensons que dès l'heure présente des microprocesseurs de seconde génération (dont la vitesse dépasse 100.000 instructions — élémentaires — par seconde) munis de périphériques lents et adaptés tels que cassette ou disque souple, lecteurs de cartes bon marché comme on en trouve maintenant, et imprimantes lentes à des prix également faibles, sont capables de réaliser des travaux de gestion à des coûts inférieurs à ceux de toute organisation centralisée utilisant des matériels importants (8).

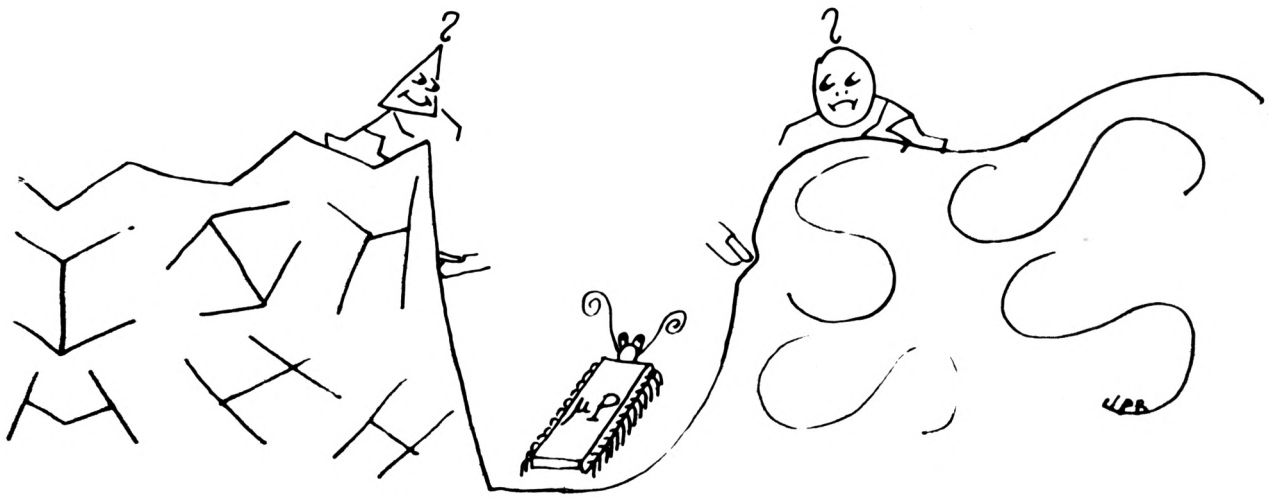
Arrivés à ce point, l'inévitable question — et nous nous la sommes posée — est la suivante : pourquoi ne sommes-nous pas déjà submergés par ces machines ?

1 - 3. JETONS LE PONT

Notre découverte, c'est que les microprocesseurs sont tombés dans une crevasse. La crevasse qui

(7) Voir le rapport de Fred Littrell dans le numéro de décembre 1974 de *L'Informatique nouvelle*.

(8) Pour une analyse en profondeur, voir «*La micro-informatique*» par B. Lussato, J.-P. Bouhot et B. France-Lanord, Editions d'Informatique, fin 1974.



sépare deux mondes clivés : celui des informaticiens et celui des électroniciens.

Au sein de notre petit univers technologique, nous nous trouvons en face d'un phénomène renouvelé de C.P. Snow dont le célèbre essai, *Les deux cultures*, mit si bien en évidence le divorce entre les communautés littéraire et scientifique. Mais cette fois-ci, il s'agit du hard et du soft.

Ouvrons un numéro récent de l'excellente revue *Electronics* (de notre grand confrère McGraw Hill) et nous lisons des lignes (citation synthétique de mémoire) de la teneur suivante, s'adressant aux électroniciens : «Jusqu'ici, nous avons réalisé nos logiques combinatoires et séquentielles au moyen de centaines de boîtiers TTL pour réaliser nos automatismes et appareils divers, or, un autre moyen — remarquablement puissant et économique — apparaît : l'utilisation d'instructions, formant un programme que l'on écrit pour un microprocesseur. Il va falloir apprendre cette technique toute nouvelle, radicalement différente».

Quant aux informaticiens, les mots qui font peur aux électroniciens, bits, octets, adressage indirect, interruptions, registre pointeur de pile, opérande immédiat, index, assembleur, leur sont au contraire familiers. Par contre, quid de la compatibilité MOS canal n et TTL ? Quid du registre adresse mémoire ROM, de l'horloge à quatre phases et des résistances de pull-up ? Quid du MOS dynamique, des sorties statiques à trois états et du latch externe ? Les buffers sont-ils prévus ou faut-il les faire en TTL ? Est-ce que le C-MOS vaut la peine d'être utilisé ?

Nous allons donc jeter le pont entre les deux domaines, revenant bien loin en arrière — mais en chaussant nos bottes de sept lieues. Rassurez-vous, il n'y aura ni statistique de Fermi-Dirac, ni zones de Brillouin, ni mécanique quantique. Comme d'habitude, nos maths ne dépasseront guère la règle de trois, et le texte sera franchement didactique : il suffira de lire attentivement et de réfléchir un peu. Le micro-ordinateur sera au bout du chemin.

CHAPITRE 2
TECHNO-LOGIQUE

Ainsi avons-nous baptisé notre activité qui va marier composants et logique binaire tout au long du trajet. Les composants, ce sont essentiellement des composants à l'état solide, fabriqués à l'aide de semi-conducteurs (silicium, germanium), et que par un abus de langage courant on appelle eux-mêmes semi-conducteurs. C'est au niveau des notions de base de l'état solide qu'il nous va falloir démarrer, en passant ensuite aux diodes, transistors et circuits intégrés de natures diverses : ce seront nos briques de construction élémentaires, à relier ensemble en fonction du schéma logique et des compatibilités électroniques.

2 - 1. PREMIER ACTE : LA DIODE ET LE TRANSISTOR

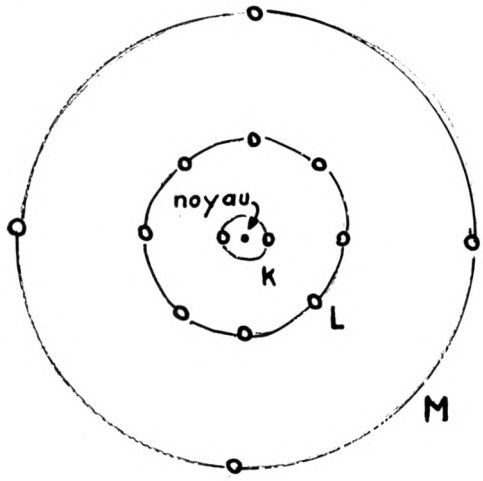
Il m'en souvient comme si c'était hier : la nuit était déjà tombée, et la brise de mer fouettait contre les volets le fil de l'antenne clandestine allant du rebord du toit jusqu'à mon lit. La veille on avait fêté mes dix ans, et je m'étais couché tard : j'aurais dû dormir déjà depuis une heure. Un peu ensommeillé, à la lumière d'une lampe électrique cachée sous les couvertures, tenant la galène dont l'achat avait fortement entamé mes économies, je cherchais son point sensible d'une main un peu tremblante. Le mince fil métallique spiralé s'accrochait aux facettes du cristal gris plombé, scintillant dans son petit tube de verre. Et soudain, le miracle : une musique lointaine et grésillante jaillit dans les écouteurs, mon cœur tressauta dans ma poitrine, mon premier poste de radio marchait...

Moins de dix ans après, Bardeen et Brattain inventaient le transistor, descendant direct de cette mystérieuse galène utilisée tout juste comme jouet sans que personne ne sache bien comment elle fonctionnait. Leur invention allait bouleverser la radio et toute l'électronique. Encore dix ans, et ce serait le tour de l'informatique.

2 - 1 - 1. Pourquoi le fer est-il froid ?

Démocrite l'avait pressenti, la matière est constituée d'une myriade de corpuscules quasi insécables — d'où le nom qu'il leur donna, *atomes* — identiques entre eux s'il s'agit d'un corps pur. Plus de deux millénaires plus tard, cette hypothèse devint un fait d'expérience. Lord Rutherford proposa le modèle planétaire d'atome qui nous

est devenu si familier, perfectionné par Niels Bohr (ne figure-t-il pas justement dans les emblèmes de nombreuses sociétés d'électronique) et nous pûmes enfin répondre à la question : pourquoi le fer est-il froid ?



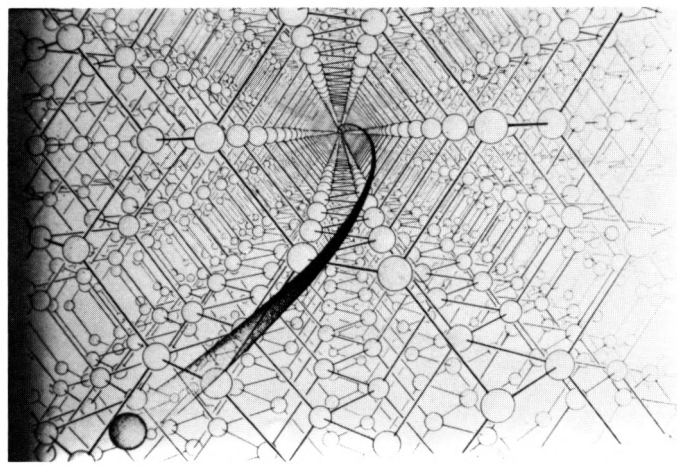
l'atome de silicium Si

- 14 électrons sur les couches KLM
- on en trouve partout sous forme d'oxyde : la silice SiO_2, sable..

Notre croquis rappelle qu'un atome est composé de deux parties : le noyau, minuscule, concentrant la quasi-totalité de la masse de l'atome, et le nuage électronique qui l'entoure. Grossissons mille milliards de fois notre atome. Le noyau, c'est une framboise dont certains grains sont des neutrons et d'autres des protons : ces deux particules sont fort semblables, à la différence près que le neutron — comme son nom l'indique — ne porte pas de charge électrique, tandis que le proton porte une charge unitaire positive (par charge unitaire, nous entendons le grain élémentaire d'électricité).

Les électrons, à peu près de la même taille qu'eux, tournoient en nuage à une bonne centaine de mètres : ils sont en nombre égal à celui des protons du noyau, et comme chacun d'eux porte une charge unitaire négative, l'ensemble de l'atome est électriquement neutre. C'est du reste la force d'attraction électrostatique entre charges de signes contraires qui remplace ici la pesanteur, bien trop faible pour conserver la cohésion de notre système planétaire microscopique en contre balançant la force centrifuge.

Si notre dessin est à deux dimensions, en fait l'atome est, lui, différent du système solaire, en ce sens que les électrons n'évoluent pas simplement dans un plan mais dans des coquilles sphériques imbriquées comme les couches d'un oignon. Chaque couche peut contenir un nombre maximum d'électrons, d'abord plus grand quand elle est plus loin du noyau : 2, 8, 18, 32, puis des nombres variables plus faibles. Quand on progresse de l'élément le plus simple, l'hydrogène (un électron) jusque vers les élémens les plus lourds, tel l'uranium (92 électrons), les couches se remplissent progressivement d'électrons en partant du centre : la dernière est donc en général incomplète. Ce sont les électrons de cette dernière couche qui interviennent seuls dans les propriétés chimiques et optiques courantes de la matière (on les appelle *électrons de valence*), les autres étant plus fortement attirés par le noyau parce que plus près.

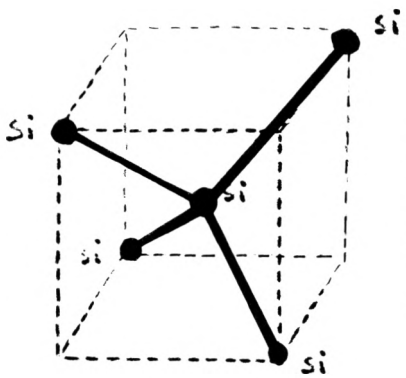


Dans un cristal, les atomes au moment de la cristallisation, sous l'action des forces électrostatiques, se rangent en un réseau tridimensionnel régulier. Leurs électrons externes sont mis en commun avec les atomes voisins. Dans une catégorie particulière d'éléments, les métaux, la dernière couche ne comprend qu'un, deux ou trois électrons de valence très faiblement liés qui peuvent se détacher et circuler dans le cristal tel un gaz électronique intersticiel : ce sont les *électrons de conduction*. Voilà donc pourquoi les métaux sont conducteurs. Accessoirement, quand vous touchez un morceau de fer, l'élévation de température que vous provoquez fait vibrer localement plus fort sur place les atomes, et les électrons «gazeux» qui circulent, entrant eux-mêmes en vibration, s'en vont rapidement répartir au loin la chaleur de votre main, la pompant ainsi avidement... et le fer paraît froid.

2 - 1 - 2. Porteurs et jonction

Entre les corps conducteus (métaux) et les isolants, figure une classe assez mal définie de corps appelés semiconducteurs. Deux d'entre eux, le germanium (Ge) et le silicium (Si), y jouent un rôle de premier plan.

Dans un cristal de silicium, chaque atome du réseau cristallin est entouré de quatre proches voisins avec lesquels il met en commun les quatre électrons les plus extérieurs de son nuage électro-



maille élémentaire du réseau cristallin du silicium

nique, les électrons de valence (chaque atome de Si se trouve donc entouré d'une couche extérieure de 8 électrons, configuration particulièrement stable qui s'appelle... un *octet* ! Coïncidence significative, aurait dit C. G. Jung).

Ces électrons extérieurs sont ceux qui permettraient normalement la circulation d'un courant à travers le cristal s'ils pouvaient se détacher. S'ils restent liés, le cristal (que nous supposons uniquement composé d'atomes de silicium c'est-à-dire absolument pur) devrait donc être isolant. En fait, sous l'influence des vibrations atomiques causées par la température, de temps à autre un électron est délogé de sa place et circule dans le cristal en devenant une charge négative libre. Ce faisant, il laisse sur place un *trou* qui se comportera comme une charge positive. Si un électron de l'atome voisin, attiré par ce trou, vient s'y loger, le trou se déplacera d'un cran dans le cristal (comme dans le jeu de taquin) ; électrons et trous, appelés

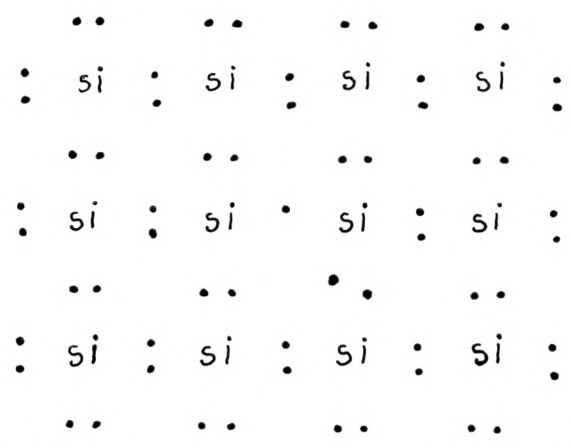
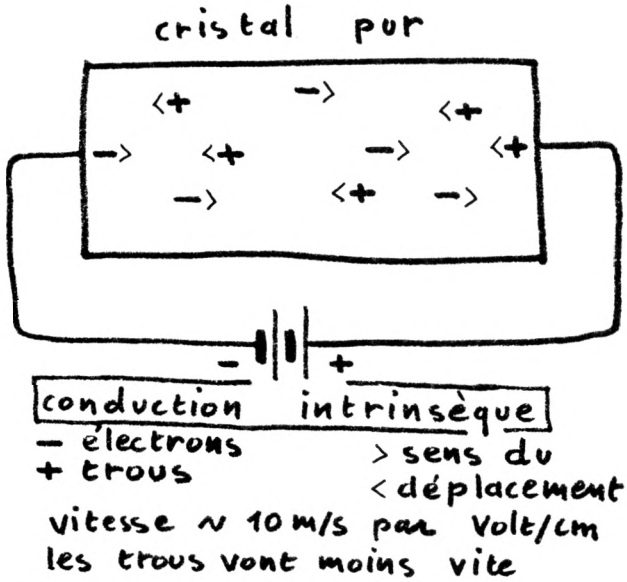


Schéma plan d'un cristal de Si. Vers le centre, un électron fait un saut NW pour combler un trou. Ce dernier sautera donc au SE d'un cran.

porteurs, sont donc créés thermiquement par *paires* à un taux augmentant très vite avec la température : il double à peu près tous les dix degrés.

De temps à autre un des électrons libres, dans le cours de son trajet aléatoire en zig-zags, vient à rencontrer un trou, et il y a recombinaison avec disparition d'une paire. Ce double processus de création et de disparition fait qu'une concentration moyenne de porteurs, dépendante de la température, s'établit dans le cristal. Si nous connectons les pôles d'une pile aux deux extrémités d'un cristal, les trous se dirigent vers le pôle négatif et les électrons vers le pôle positif, au fur et à mesure de la création des paires. Un faible courant parcourra le cristal, qui est donc finalement non point un isolant comme nous l'avions prévu initialement, mais un conducteur à résistance élevée qu'on appellera donc *semiconducteur* : le mécanisme de conduction que nous venons de détailler s'appelle *conduction intrinsèque*.



Introduisons maintenant en faible quantité (un milliardième à un millionième) dans du silicium en fusion une impureté, par exemple de l'arsenic, dont les atomes ont une taille très voisine de ceux de Si. A la cristallisation, les atomes d'arsenic répartis au hasard viendront chacun prendre la place d'un atome de silicium dans le réseau cristallin. Mais l'arsenic (9) possède cinq électrons dans sa couche électronique extérieure (on dit qu'il est *pentavalent*) : quatre seront liés aux atomes voisins, et le cinquième pourra aisément circuler dans le cristal. Un tel atome est *donneur*.

Le nouveau semiconducteur — on dit qu'il est *dopé* — laissera passer beaucoup plus facilement le courant que le silicium pur, l'essentiel de la

(9) Le phosphore et l'antimoine peuvent remplir le même rôle que l'arsenic.

conduction étant assuré par les électrons excédentaires de l'arsenic que l'on qualifie de *donneur*. Vu le mode de conduction par électrons, ce type de semiconducteur est désigné par N, ou n (pour *négatif*). La sensibilité du procédé est très grande : un dopage avec un atome d'antimoine pour dix millions de silicium multiplie la conductibilité de ce dernier plus de cent fois. Il est à remarquer que les atomes d'arsenic dépouillés de leur cinquième électron externe constituent maintenant des charges positives fixes ancrées dans le cristal, au contraire des trous mobiles. Le phénomène de conduction intrinsèque se superpose évidemment à la conduction N et n'a pas été représenté sur la figure. Comme il y a maintenant beaucoup plus d'électrons que de trous, ces derniers seront appelés *porteurs minoritaires,* les *porteurs majoritaires* étant bien entendu les électrons.

Une interprétation un peu différente de ce phénomène de conduction, que nous trouverons fort utile par la représentation du fonctionnement des transistors, est donnée par le principe de *neutralité de charge d'espace*. A tout moment, l'ensemble du cristal ne peut que rester sensiblement électriquement neutre, pour des raisons d'attraction électrostatique globale puissante : c'est l'inclusion des charges positives fixe de l'arsenic qui permet l'existence d'un nombre égal d'électrons libres circulant dans le cristal, électrons dont la provenance peut être entièrement extérieure (pôle négatif de la pile dans le cas de la figure).

Symétriquement, nous aurions pu ajouter à du silicium, au lieu d'arsenic, par exemple du gallium (10) trivalent qui ne possède que trois électrons sur sa couche externe : les atomes de gallium capturent un quatrième électron destiné à assurer la liaison avec les atomes voisins, acquérant ainsi une charge négative, ce qui permettra l'existence de trous. Un tel atome est dit *accepteur*. Le cristal obtenu par cette introduction d'impureté est cette fois du type P, ou p (pour *positif*) et la conduction s'y fait par trous : le déplacement des trous, pour un même sens de parcours du courant, se fait en direction opposée à celle du déplacement des électrons dans les semiconducteurs de type N.

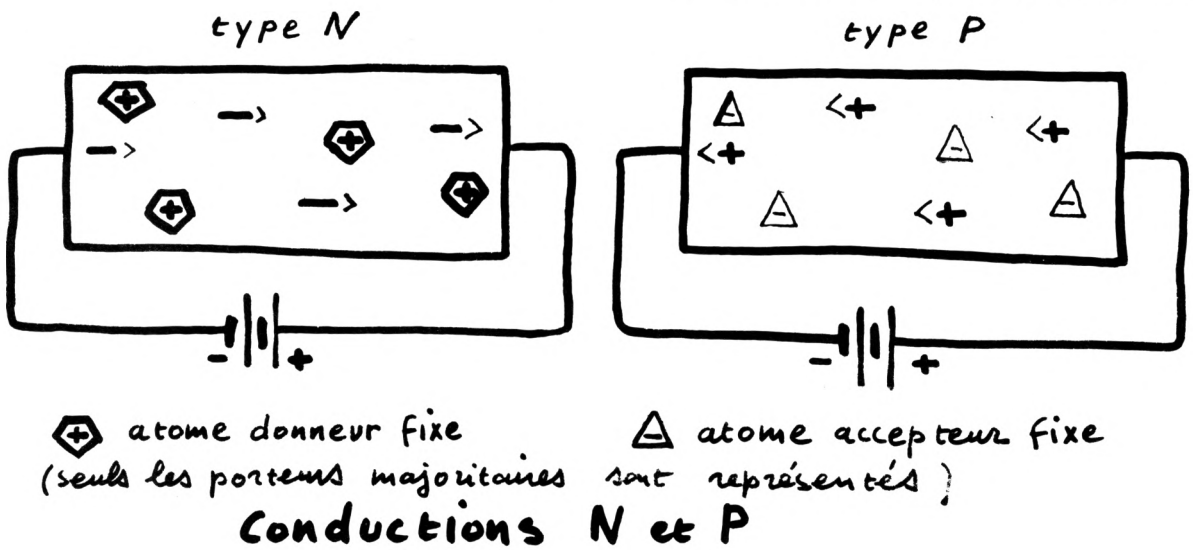
Le mystère des semiconducteurs étant maintenant éclairci, nous pouvons nous attaquer à deux des composants essentiels de tout ordinateur, la diode et le transistor, qui sont aussi les éléments constitutifs de base des circuits intégrés.

2 - 1 - 3. La diode à jonction

Supposons maintenant que nous ayions réussi, par un processus que nous verrons dans le cas des circuits intégrés, à juxtaposer dans un même cristal de silicium (ou de germanium) deux zones de type N et P par un dosage convenable. Connectons le pôle négatif d'une pile à la région P et le pôle positif à la région N. Une telle combinaison de connexions soumet le cristal (qui constitue une *diode*, comme nous allons le voir) à une *polarisation* dite *inverse*. Que va-t-il se passer ? Electrons et trous, attirés respectivement par les connexions positives et négatives, refluent vers les extrémités de la diode en s'écartant de la jonction autour de laquelle se crée une mince couche vide de porteurs, dite *couche d'arrêt*. Ce reflux se stabilise lorsque les attractions sur les porteurs dues aux atomes d'impureté inclus dans les zones N et P équilibrent l'attraction des connexions positives et négatives.

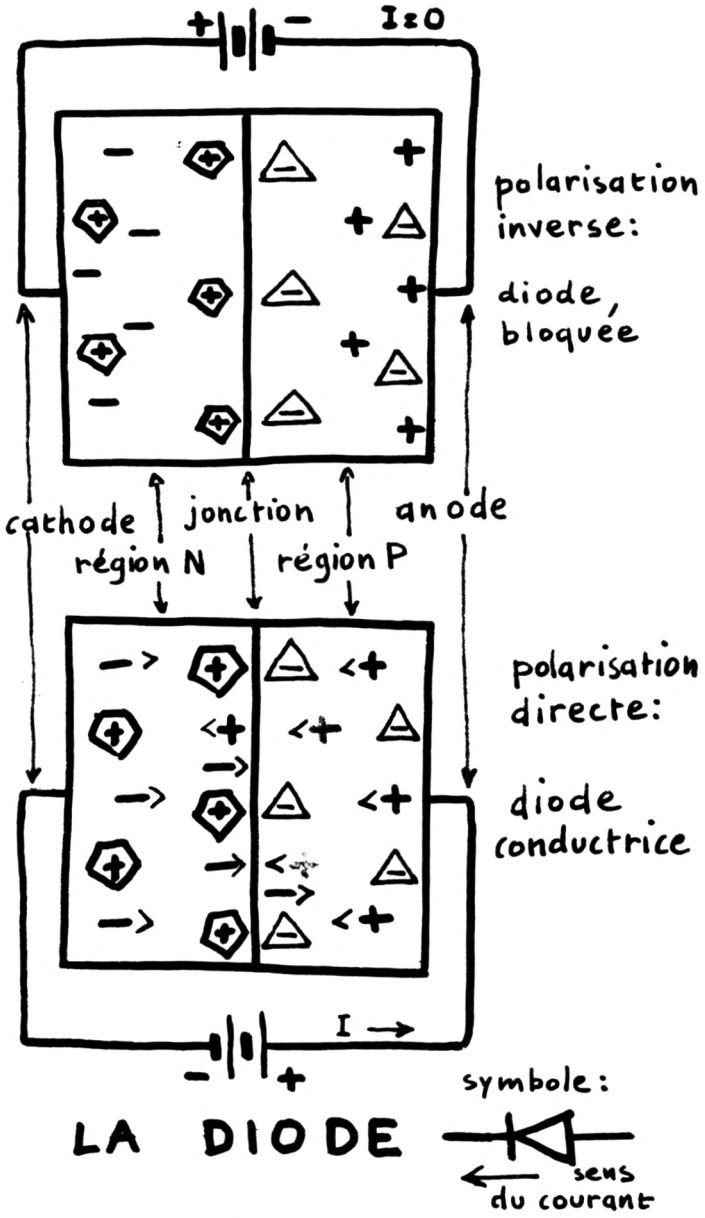
Aucun courant ne passe donc, aucun porteur ne pouvant atteindre la jonction. On dit que *la diode est bloquée*. Seul un très léger *courant de fuite* dû à la création thermique de paires au voisinage de

(10) Le bore, l'aluminium ou l'indium peuvent remplir le même rôle que le gallium.



Conductions N et P

la jonction se manifeste (11). Si l'on augmente régulièrement la tension de la pile, ces électrons créés thermiquement se déplacent à vitesse de plus en plus grande dans le cristal, jusqu'au moment où leur énergie devient suffisante pour qu'ils arrachent d'autres électrons aux atomes du semiconducteur. Leur nombre se multiplie alors rapidement : il se produit un *effet d'avalanche* soudain, et la diode devient brusquement conductrice laissant passer une très grande quantité d'électrons. Elle sera très rapidement détruite par échauffement si le circuit de la pile ne comprend pas une résistance limitatrice. La tension à laquelle se produit ce phénomène, utilisé dans certains circuits, se nomme *tension d'avalanche*, ou *tension inverse de claquage*.



Connectons maintenant la pile à la diode dans l'autre sens, correspondant à la *polarisation directe*. Cette fois trous et électrons, repoussés des extrémités, se dirigent vers la jonction au voisinage de laquelle ils se combinent. Simultanément, le fil négatif injecte de nouveaux électrons dans la zone N, et le fil positif, arrachant par attraction les électrons des atomes du semiconducteur placés au voisinage de sa connexion, injecte en fait des trous dans la zone P. Un courant important circule donc à travers la diode, même avec une faible tension appliquée. La jonction est en fait le siège de phénomènes très complexes : une partie de l'énergie provenant des recombinaisons trous-électrons peut s'évader sous forme de photons, et elle émettra une lumière rouge ou bleue pour une polarisation directe ou inverse dans le cas de diode au phosphure ou à l'arséniure de gallium. C'est le principe des *LED (diodes électro-luminescentes, light-emitting diodes)* employées par toutes les calculatrices de poche, et que nous aurons l'occasion d'utiliser. Développant cette idée, les chercheurs de General Electric, il y a une douzaine d'années, fabriquèrent la première *diode laser*, semiconducteur révolutionnaire constituant un laser à excitation électrique et non plus lumineuse, sans doute élément clé des futurs systèmes de communication à optique fibrée.

Mais revenons à notre diode classique. Voilà donc réalisé un clapet à électrons de fonction identique à la diode à vide qu'utilisaient les premières calculatrices, mais quelle différence ! Plus besoin de courant de chauffage, d'ampoule où l'on fait le vide. De plus, la résistance étant très faible, un élément semiconducteur gros comme une tête d'allumette suffit pour les applications les plus courantes où il remplace un tube de plusieurs centimètres. Il a enfin le gros avantage de bien fonctionner avec des tensions très faibles.

Cette diode, c'est la version moderne de l'antique galène, ce cristal de sulfure de plomb sur lequel on appliquait l'extrémité d'un fil métallique. Les premières diodes, du reste, étaient constituées par un fil métallique soudé sur un semiconducteur dopé, et au moment de la soudure une deuxième zone de type différent se formait autour de la pointe du fil. Ce type de diode a été longtemps utilisé dans les ordinateurs, pour lesquels est fondamentale la rapidité de cessation de passage du courant lorsqu'on applique brusquement, par l'arrivée d'une impulsion, une polarisation inverse après une polarisation directe. La diode à jonction dont nous venons de voir le fonctionnement n'a fait son apparition qu'après : elle existe depuis des tailles minuscules jusqu'aux gros

(11) Si un faisceau lumineux tombe sur la jonction, les *photons* du faisceau (grains de lumière qui sont de l'énergie pure) heurtent violemment les électrons des atomes, en délogent quelques uns, et le courant croît brusquement : c'est la *photo-diode*. Il existe de même des *photo-transistors*.

modèles industriels laissant passer des centaines d'ampères...

De même, lorsque Bardeen et Brattain inventèrent le transistor en 1948, leur première réalisation était à deux pointes (distantes d'un centième de millimètre), et assez imparfaite. Mais peu de temps après, le Dr W. Schockley généralisait le principe du transistor à jonction, aujourd'hui présent dans tous les postes portatifs du monde aussi bien que dans les plus récents ordinateurs, satellites ou véhicules interplanétaires.

A vrai dire, cette invention représente par rapport à la diode un élément nouveau dans cette espèce de «cristallisation» (c'est bien le cas ici !) de notions disparates sur les semiconducteurs en usage depuis longtemps sur une base empirique : les redresseurs à oxyde de cuivre, les protections de lignes téléphoniques contre la foudre par des pastilles de céramique au carbure de silicium en usage vers les années 1930... Un fil de cuivre enfoncé dans une pomme de terre permet de faire une diode (pas très bonne, car utilisant une autre classe de semiconducteurs, les semiconducteurs ioniques : je fus tout de même obligé d'acheter ma galène...)

Le transistor, c'est l'intrusion brutale de la physique des solides dans l'électronique : sa théorie complète remplit des centaines de pages de calculs de mécanique quantique. Mais nous laisserons les fonctions d'ondes de côté : diode à jonction et porteurs qui nous sont maintenant familiers suffiront à élucider l'essentiel du mystère.

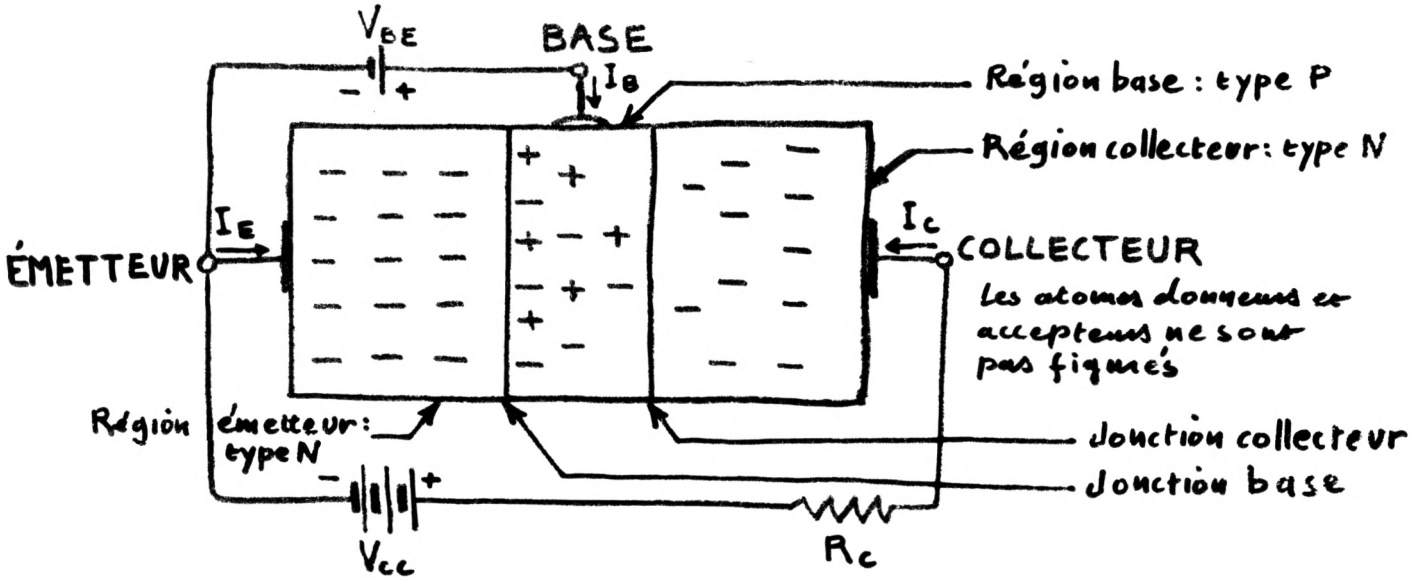
2 - 1 - 4. Le transistor à jonction

C'est un sandwich de semiconducteurs. Si le «pain» est de type P, le «jambon» sera de type N : on aura dans ce cas un transistor PNP. Il existe de même des transistors NPN, et c'est ce dernier type que nous prendrons comme exemple. Sur le schéma qui le représente, nous n'avons figuré que les porteurs, maintenant que vous êtes familiarisés avec les atomes donneurs et accepteurs fixes ancrés dans le réseau cristallin. La zone intermédiaire s'appelle *base*, l'une des zones extrémités *émetteur* et l'autre *collecteur*. Notre transistor a donc trois fils.

Nous nous trouvons en fait devant deux diodes accolées en sens inverse, pourvue chacune de leur jonction. Supposons tout d'abord que nous connections entre émetteur et collecteur une pile dans le sens indiqué : la jonction émetteur, polarisée dans le sens direct, ne s'opposerait pas au passage du courant, mais la jonction collecteur, elle, est polarisée en sens inverse et bloque le passage du courant.

Notre transistor est donc isolant (quel que soit d'ailleurs le sens de la pile). Ajoutons maintenant une deuxième pile, entre base et émetteur, qui polarisera dans le sens direct la jonction émetteur.

Cette fois, un flot d'électrons se dirige de l'émetteur vers sa jonction et la traverse pour se recombiner avec les trous venant en sens inverse, mais la base étant très mince, (inférieure au centième de millimètre), dès qu'ils ont traversé la première jonction les électrons sont attirés par le collecteur plus fortement positif, et s'y précipitent au lieu d'aller vers la connexion de la base. Seule



LE TRANSISTOR est un sandwich de silicium (pâtes sablées aux deux parfums)

une faible portion d'entre eux réussit à se recombiner : un courant très faible circule donc entre base et émetteur, tandis qu'un courant beaucoup plus fort va de l'émetteur au collecteur (12), sa présence n'étant rendue possible que par l'établissement de ce faible courant base-émetteur qui le contrôle.

La notion de charge d'espace permet de voir sous un autre angle ce principe de fonctionnement du transistor. Mettons en contact un bref instant la connexion de la base avec le pôle positif de sa pile : il se produit une injection de trous dans la base, et l'existence de cette concentration de trous supplémentaire permet la présence d'un nombre équivalent d'électrons circulant dans la base en provenance de l'émetteur, donc la circulation d'un courant entre émetteur et collecteur à travers la base qui cesse d'être isolante. Notre transistor est en fait un véritable interrupteur qui peut basculer (nous dirons *commuter*) en un temps très court sous l'action d'un faible courant : c'est là son utilité principale en informatique. Théoriquement, le courant circulerait ainsi indéfiniment sans que le fil de base soit connecté : en fait, quelques électrons se recombinent avec des trous en les annihilant, et il est nécessaire d'en fournir un léger apport constant en connectant le fil positif de la base.

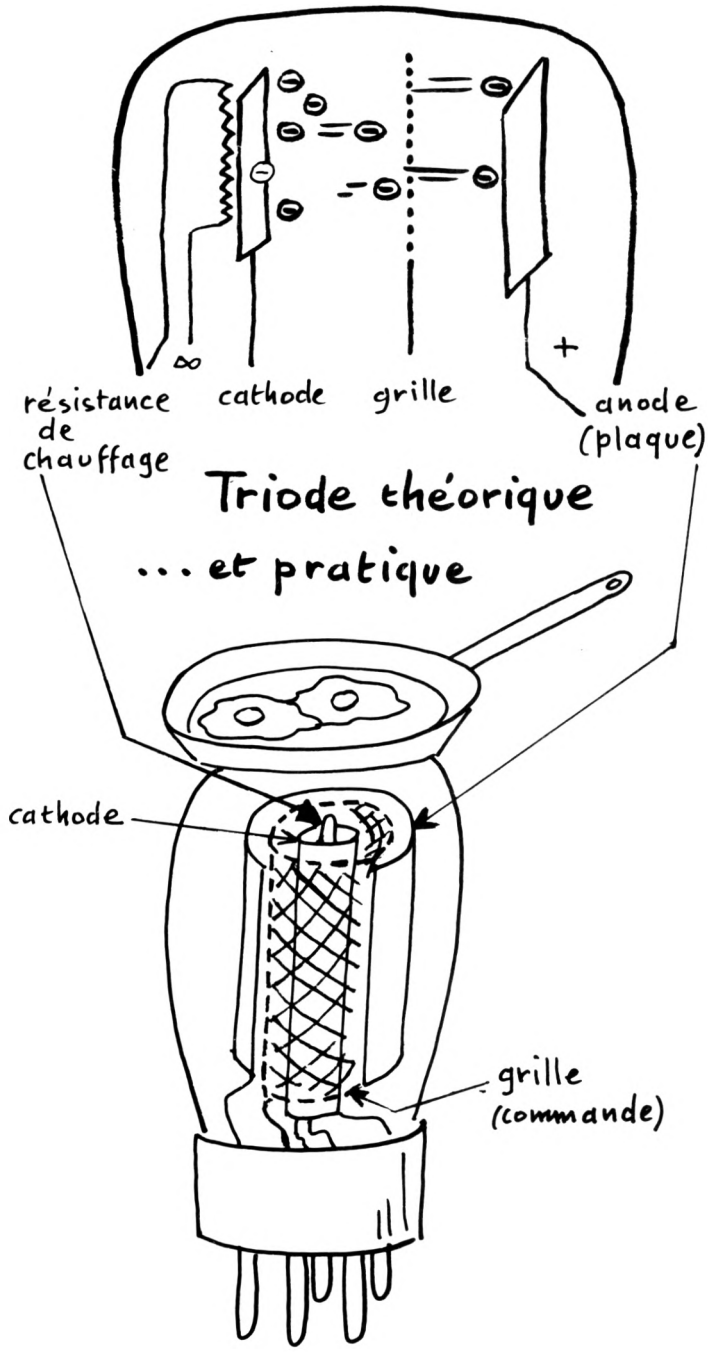
Bilan général : on peut contrôler un fort courant à l'aide d'un autre beaucoup plus faible, d'une façon tout à fait analogue à ce qui se passe avec une triode, donc obtenir un effet d'amplification. Mais quels progrès sur les tubes à vide pour nombre d'applications !

Comme pour la diode à jonction, disparition du chauffage, donc du dégagement de calories et de l'absorption de puissance électrique (1/2 à 5 watts pour les tubes), réduction de la taille à celle d'un petit pois, et surtout grosse diminution des tensions d'alimentation qui sont maintenant de l'ordre de la dizaine de volts et non plus des centaines de volts nécessaires aux tubes à vide des ordinateurs de première génération, où l'on doit attirer les électrons suffisamment pour leur faire franchir l'espace cathode-plaque.

C'est donc dans le domaine des appareils de surdité (amplificateur caché dans des branches de lunette par exemple) que les transistors ont fait leur toute première apparition, vite étendue à tous les domaines où légèreté, miniaturisation et faible consommation sont de première importance : postes de radio et d'autres appareils portatifs, satellites alimentés par de parcimonieuses piles solaires, etc.

Vous avez sans doute tous entendu raconter un jour ou l'autre l'histoire fantaisiste de cet homme qui ayant une dent en or contre un plombage pré-

(12) Nous rappelons que le sens conventionnel du courant est inverse de celui du flot des électrons.



La résistance chauffe la cathode et fait "bouillir" les électrons qui sont alors attirés par la plaque : ils franchissent le vide (pas de molécules d'air gênantes).

Si la grille est rendue négative, elle les bloque et coupe le courant avec une dépense d'énergie minuscule. C'est le principe même du type de transistor le plus employé pour les micro-processeurs, qui est une véritable triode à l'état solide. Nous l'étudierons en détail.

tendait entendre Radio-Luxembourg quand il touchait les barreaux de fer de son lit ! Plus fort encore, mais vrai, des dentistes ont mis au point un bridge émetteur qui par une émission en morse (sûr !) permet de mesurer la surface de contact des dents pour améliorer l'appareil !

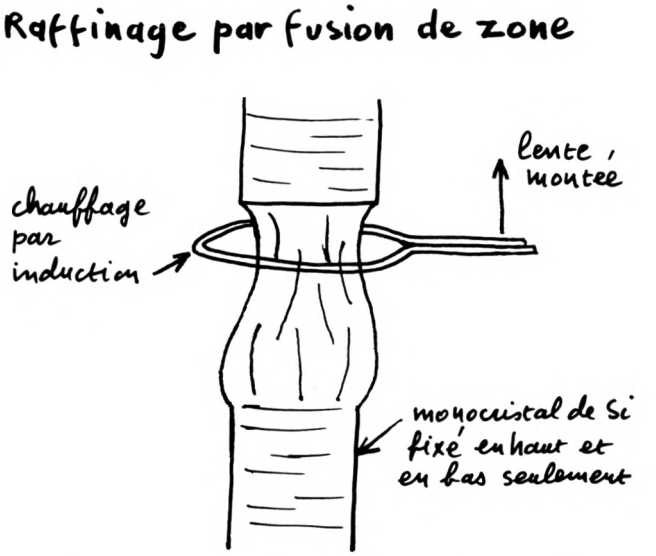
Aujourd'hui, les transistors sont partout. Ordinateurs terrestres ou embarqués, appareillage aérien radio et radar, postes de TV portatifs, profitent de leur potentiel de vie théoriquement illimité. Autrefois, le dépanneur commençait par vérifier les tubes. Maintenant, le transistor est le dernier élément qu'il inspecte.

Dans le domaine des schémas électroniques, la présence des transistors complémentaires PNP et NPN permet de réaliser des montages inconnus avec les tubes, qui ne peuvent évidemment fonctionner avec une plaque négative !

Les premiers transistors, fabriqués dans des semiconducteurs polycristallins, étaient improductibles deux fois de suite avec les mêmes caractéristiques. Il a fallu développer concurremment des techniques d'obtention de monocristaux importants et de raffinage par fusion de zone (inventé par W.G. Pfann) qui a permis de descendre le taux d'impuretés étrangères avant dopage au dix-milliardième nécessaire. En 1955, introduction du silicium qui, permettant de fabriquer des transistors résistant beaucoup mieux aux élévations de température et se prêtant très bien aux fabrications automatisées, devait à peu près entièrement détrôner le germanium : sur une tranche de cristal plate d'un poli optique, à cette époque grande comme un timbre poste, on fabriquait déjà en un seul processus 400 transistors de deux dixièmes de millimères sur trois, les ouvrières soudant les fils sur des éléments à peine visible à l'œil nu, suivant l'opération sur écran de TV, et marquant au micromanipulateur les transistors que les tests révélaient défectueux.

Puis ce fut une longue série de perfectionnements, à la recherche de procédés permettant de fabriquer les jonctions les plus fines et les plus planes possibles : transistors *drift, mesa* (rongés par des acides comme par l'eau de pluie et le vent ces «mesa», formations géologiques en table qu'on voit en toile de fond de tous les westerns)... A chaque fois, un petit progrès nouveau. Et enfin les chercheurs de Fairchild inventèrent le procédé planar : le principe, toujours indéfiniment perfectionné dans le monde entier, source de tous les procédés actuels, avait vu le jour. Le circuit intégré monolithique était en vue.

Mais ne brûlons pas les étapes au risque de faire une salade de semiconducteurs, il nous faut voir maintenant l'utilisation du transistor dans l'ordinateur.



2 - 2. AU MITAN DES MILLE PORTES

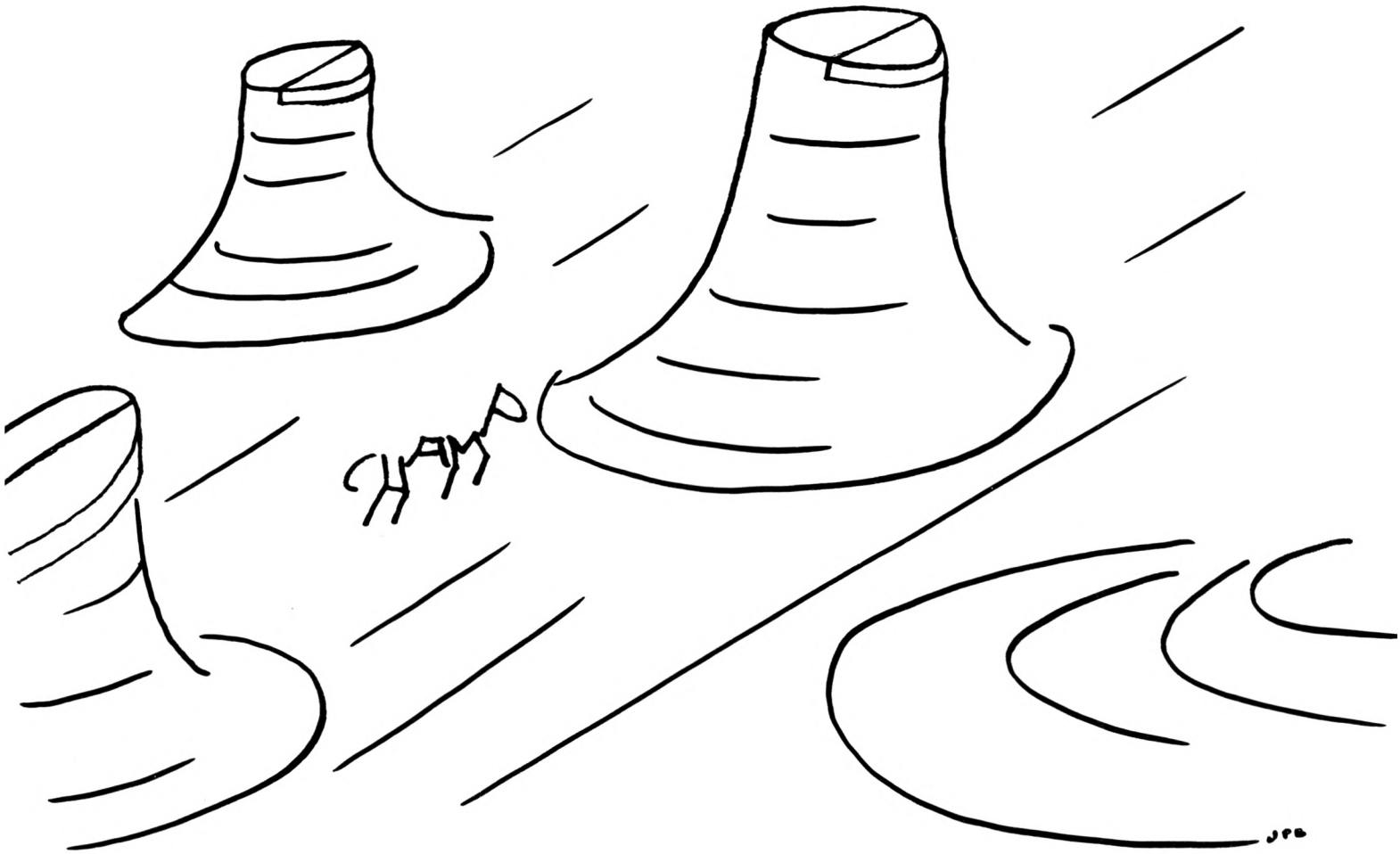
2 - 2 - 1. Il faut qu'une porte soit ouverte ou fermée

Pendant que Démocrite imaginait des atomes de matière, d'autres Grecs cherchaient des atomes, abstraits ceux-là, des atomes de pensée : les lois fondamentales de la logique. Toute activité humaine, intellectuelle ou non, repose sur quelques conventions universelles dont d'adoption entraîne des conséquences inéluctables : si je décide d'appeler *deux* l'unité augmentée d'elle-même, je ferais bien de ne pas l'appeler *trois* le printemps prochain, sous peine de perdre les avantages que me procurait une définition, même intuitive, des nombres entiers. De même si j'attribue la caractéristique «*mortel*» à l'être humain, et si l'on me concède sans trop de peine que Socrate est un être humain, je dois en conclure que Socrate est mortel, sans être tenu de le vérifier expérimentalement par la cigüe.

Dans le premier cas (deux = un + un), une définition parfaitement arbitraire m'impose une certaine cohérence avec moi-même si je veux utiliser ma définition à quelque chose de concret, compter des pièces de monnaie par exemple. Dans le second cas (Socrate est mortel), une constatation universelle entraîne une propriété que je ne puis nier sans remettre en cause la constatation initiale : une sorte de «loi de nature» s'oppose à ma liberté ultérieure.

Ce sont ces contraintes, difficilement justifiables si ce n'est par la forme même de notre esprit, que les Anciens cherchaient à ramener au minimum minimorum, au plus petit noyau possible : la démarche se compare évidemment à l'hypothèse des atomes sur le plan matériel.

Il appartenait à notre siècle de réaliser la synthèse : la logique par les atomes. C'est là qu'intervient le transistor, ce petit contrôleur élémentaire qui autorise ou interdit le passage d'un courant et qui va se trouver investi d'une fonction gigantesque : assurer le respect de la logique dans l'exécution des travaux de la machine. Plus encore, c'est lui qui va assurer l'exécution -même, tous les travaux étant ramenés au contrôle de la circulation de courants dans une multitude de voies disposées convenablement.



Auparavant, il aura fallu que toutes les activités que nous confierons à notre machine aient été exprimées sous forme de 1 et de 0 correspondant au passage ou au non-passage du courant. Pour cela notre tâche aura été bien sûr facilitée par les travaux des grands logiciens du XIXe siècle : Boole, de Morgan, Babbage, Jevons, Lewis Carroll, Venn, (13) puis par ceux des créateurs de l'informatique du XXe siècle : Turing, Post, John von Neumann, Shannon.

Le principe en est simple : les travaux intellectuels sont de deux natures ; les uns sont *arithmétiques* : additionner, soustraire, multiplier, diviser, élever au carré, prendre la racine carrée, prendre le logarithme. Or il est possible de représenter tous les nombres au moyen de 0 et de 1 : c'est le système binaire bien connu ; les autres sont *logiques* : faire quelque chose *si* a est plus grand que b, calculer le salaire net *si* le salaire brut a déjà été calculé, annoncer la victoire dans une partie d'Echecs *si* le roi adverse est mat. Or, ces opérations logiques sont conditionnées par la vérité ou la fausseté d'expressions qui ne peuvent posséder que ces deux caractéristiques : ou le roi adverse est mat, ou il ne l'est pas. Le tiers est exclu, selon l'expression célèbre d'Aristote. Donnons donc 1 pour vrai, 0 pour faux, et chargeons un transistor, contrôlé par le courant de condition, d'orienter le traitement d'après la vérité ou la fausseté qu'il observe.

Voilà pour la théorie. Mais reste à faire une addition avec des transistors, reste à enregistrer un ordre rentré au clavier de Nanostar avec des transistors, bref, reste à organiser la logique binaire. La tâche paraît lourde. Elle l'est, mais les fabricants de composants sont là pour nous la faciliter : c'est leur raison d'être. Tels les inventeurs du Meccano de notre enfance, ils vendent des éléments plus ou moins standardisés que nous assemblerons selon nos besoins. Certains de ces besoins sont très généraux : les opérations arithmétiques très simples par exemple. Ils sont satisfaits par le microprocesseur qui rend possible notre jeu de construction tout entier. D'autres sont plus spécifiques : à nous de les définir, puis d'y satisfaire par un assemblage de boulons, de longerons, d'écrous qui nous sera propre. Une différence : boulons, longerons et écrous seront des variantes plus ou moins complexes de la porte qui s'ouvre ou se ferme selon les ordres données par un transistor, ou par plusieurs dès que tout se complique.

Apprêtons-nous à ouvrir et à fermer des portes : des milliers de portes qui se commandent les unes les autres, respectant le principe d'Aristote agréablement reformulé par Musset : «*Il faut qu'une porte soit ouverte ou fermée*».

(13) Tous Anglais, à l'époque de plus grande puissance de l'Empire Britannique !

2 - 2 - 2. Du comportement de quelques portes fondamentales

L'atome de logique que cherchaient les Grecs est alors le même que l'atome d'information (14) : c'est le chiffre binaire 0 ou 1, appelé *bit*. Petit morceau anglo-saxon, (contraction de «binary digit», mais n'oublions pas que «a bit of luck» veut aussi et d'abord dire «un peu de chance»), le bit peut enregistrer l'état d'une porte : ouverte, le voici à 1 ; fermée, le voilà à 0.

Quant à la porte, le système qui la commande est lui-même une suite de bits : un ou plusieurs dont les états respectifs déclenchent l'ouverture ou la fermeture de la porte. En logique, on parle d'une *fonction logique* déterminant la valeur d'une *variable logique*. En électronique, on parle d'un *circuit logique* ou d'une *porte* (gate en anglais) déterminant un *signal* ou *état logique*. Mais le point fondamental est qu'il y a similitude totale entre la pensée logique de celui qui conçoit les séries de fonctions aptes à représenter un calcul, et la réalisation électronique concrète du circuit qui effectuera réellement le même calcul. Cette similitude remarquable a été notée vers 1930 par Claude Shannon, et c'est elle qui a autorisé la construction à grande échelle de machines logiques (15). Le point essentiel acquis ces dernières années est, nous le verrons plus loin, que les étapes intermédiaires entre la conception et la réalisation ont été considérablement réduites ; le dessin-même de la surface du microprocesseur renseigne sur ses fonctions logiques. Contrairement au célèbre principe d'action «La carte n'est pas le territoire», ici la carte théorique et le territoire pratique sont quasi-identiques (16).

Voyons donc les éléments de base du Meccano logique : ce sont les portes fondamentales de la micro-informatique (en toute rigueur, nous le verrons plus loin, on ne devrait appeler *porte* un *circuit logique* que lorsqu'il est utilisé pour laisser passer ou bloquer un signal ; en pratique nous confondrons les deux appellations).

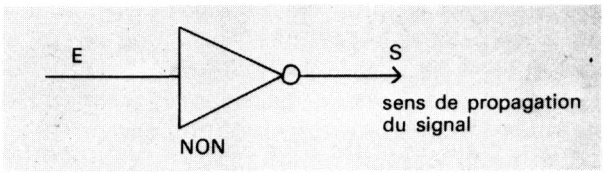
La première s'appelle NON, ou NOT en anglais. Elle transforme un signal 0 en signal 1, un signal 1 en signal 0. Systématiquement opposée à ce qu'elle reçoit, les logiciens la nomment *négation*.

(14) Pour les définitions et propriétés de l'information, ainsi que pour le détail du système binaire, cf «*Un Fil d'Ariane : l'ordinateur à la portée de tous*» J.-P. Bouhot. Editions d'Informatique. Paris 4e édition, 1979.

(15) Les mathématiciens parlent d' «isomorphisme» entre «algèbre de Boole» et «algèbre des circuits». Cet isomorphisme, découvert indépendamment de Shannon par deux Japonais, portait à l'époque sur les circuits à relais électromécaniques, ce qui ne change rien à l'affaire.

(16) «*Dans l'intérêt de la raison, n'oubliez pas : la carte n'est pas le territoire*» répété A.E. Van Vogt dans ses célèbres romans de Science-Fiction. Mais c'est normal : ardent propagandiste de la logique non-A (non-Aristotélicienne), il s'oppose (superficiellement peut-être) à notre logique informatique tout issue d'Aristote. Pour plus de détails sur le non-A, voir *L'Informatique nouvelle*, novembre 1972, Encyclopédie, J.-P. Bouhot pp. 63-64.

Les électroniciens la symbolisent par un triangle et un cercle (c'est ce dernier qui représente vraiment le symbole de la négation) :



Pour noter facilement le comportement d'une porte, le logicien emploie volontiers une table dite «*table de vérité*». D'un côté les valeurs possibles du signal d'entrée, de l'autre les valeurs correspondantes du signal de sortie.

Voici la table de vérité de la porte NON :

NON	E	S
	0	1
	1	0

E est l'entrée, S la sortie. On note souvent S = E, qu'on lit «S égale E barre», ou «S égale non-E».

La logique est souvent affaire de choix : «Je prends mon parapluie s'il pleut *ou* s'il y a des nuages vraiment trop noirs». Deux portes logiques vont traduire deux systèmes de choix : le OU inclusif et le OU exclusif.

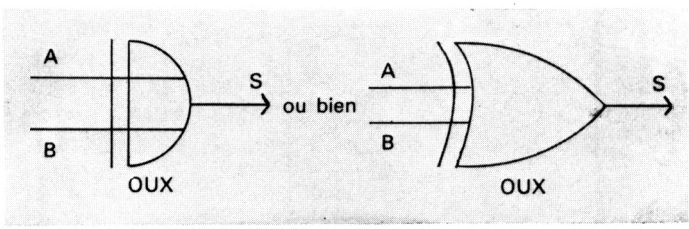
Le *OU inclusif*, ou OU tout court, ou OR en anglais, ou somme logique, donne en sortie un signal 1 si l'un ou l'autre ou les deux entrées à la fois de la porte reçoivent un signal 1.

Si A et B sont les deux entrées, S la sortie, la table de vérité du OU inclusif est la suivante :

OU inclusif	A	B	S
	0	0	0
	0	1	1
	1	0	1
	1	1	1

S est souvent notée alors : S = A + B, mais il ne faut pas confondre le signe + avec le symbole de l'arithmétique usuelle.

Sur les schémas électroniques, on trouve les représentations :



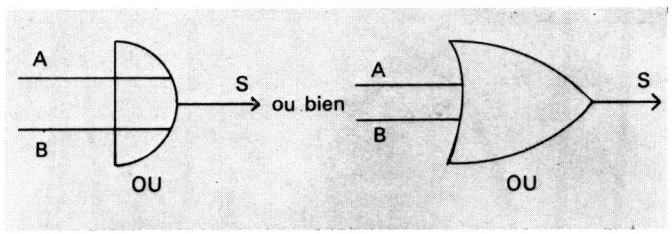
L'usage actuel nous a fait adopter la représentation en ogive dans ce qui suit, mais la normalisation est loin d'être acquise !

Le *OU exclusif*, ou OUX, ou XOR en anglais, est moins tolérant que son frère inclusif : pour le laisser passer il n'accepte qu'un seul signal 1 en entrée, et impose 0 en sortie lorsque les deux signaux d'entrée valent 1. «Il faut qu'une porte soit ouverte ou fermée», sous-entendu pas les deux en même temps, se traduit en logique par un OU exclusif. De même un feu routier est vert ou rouge, le «ou» est exclusif. (Alors que le même feu peut être rouge ou orange ou les deux dans certains pays, et donc posséder aussi une logique à OU inclusif).

La table de vérité du OU exclusif est celle-ci :

OU exclusif	A	B	S
	0	0	0
	0	1	1
	1	0	1
	1	1	0

S est souvent notée S = A + B. Les schémas électroniques sont :



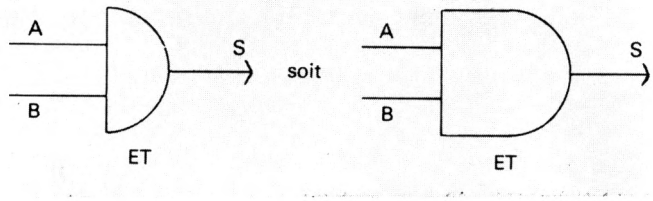
Il nous reste une porte fondamentale, ou une fonction logique de base à voir : *la fonction ET*. Pour qu'un signal traverse cette porte à deux entrées, il faut que l'une *et* l'autre des entrées reçoivent un signal 1. En logique, nous dirons : «un quadrilatère est un rectangle si c'est un parallélogramme *et* s'il possède au moins un angle droit». Si l'une des deux conditions n'est pas remplie, la proposition est fausse.

La table de vérité est alors simple :

ET	A	B	S
	0	0	0
	0	1	0
	1	0	0
	1	1	1

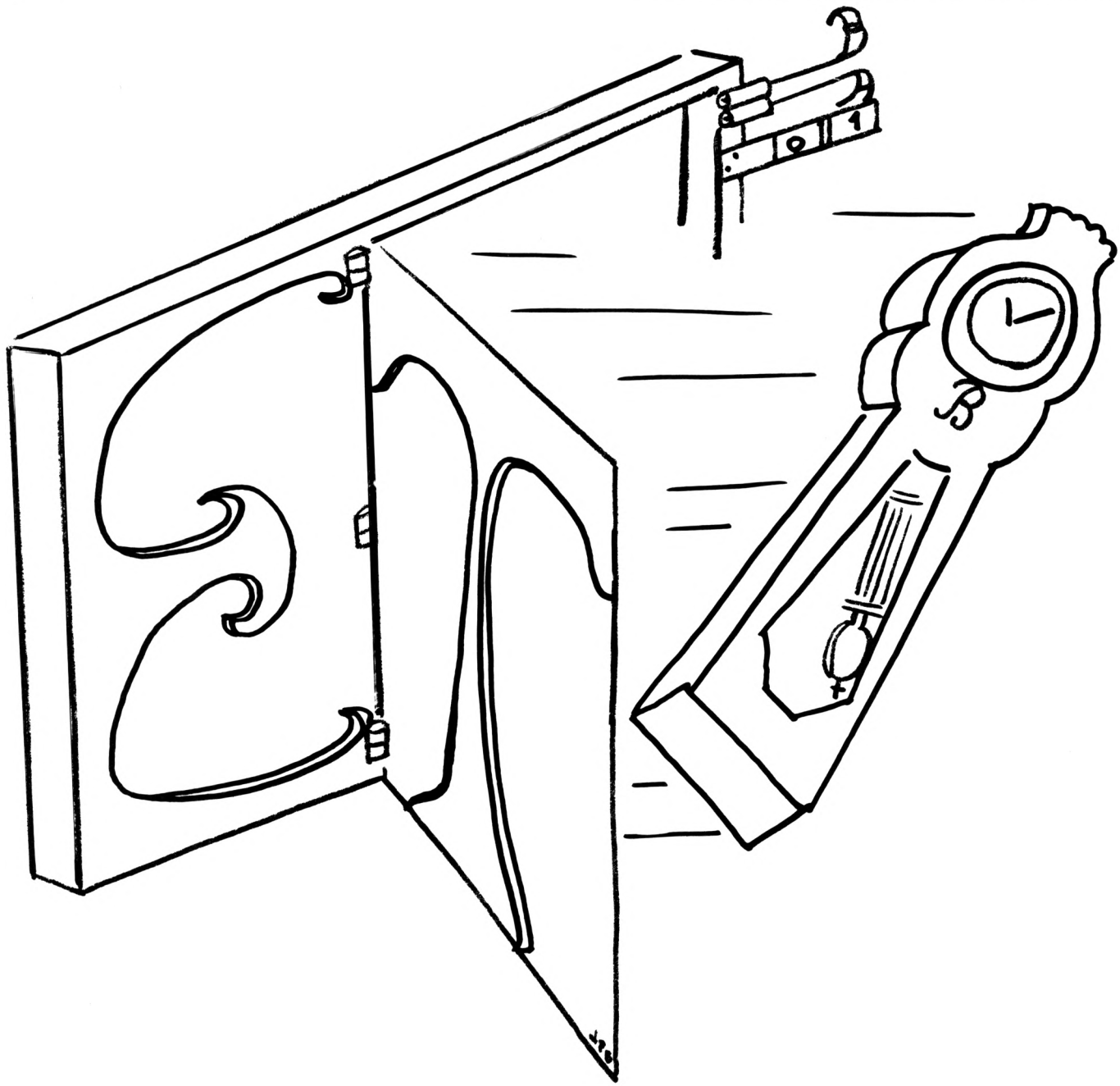
On note S = A . B, et l'on parle souvent de «produit logique» : en effet, on observe que S = A x B au sens classique de la multiplication.

Le symbole électronique correspondant est soit :



En anglais, la porte ET s'appelle AND, bien sûr.

Si nous considérons que l'entrée B correspond à un signal que nous allons vouloir laisser passer ou non, A sera le signal de commande qui *ouvrira ou fermera la porte* à B (ouverte pour A = 1, fermée pour A = 0) : c'est là une utilisation typique du circuit logique ET. Incidemment, dans le feu d'une récente discussion sur un élément de



Nanostar où B était un signal d'horloge, cette superbe phrase a volé à travers la pièce : «et alors la porte est ouverte, et l'horloge passe»...

Une figure résume en tableau les différentes représentations, schématiques ou algébriques des portes fondamentales que nous venons de voir (ci-dessous).

Observons que tout cela ressemble beaucoup à l'arithmétique élémentaire, jusque dans la notation : des « + », des «x» notés par un point, et une opération NON qui ressemble au changement de signe 1 en — 1, à cette différence près que le signe est au-dessus du nombre qu'il modifie, et non devant. Par ailleurs 1 = 0 et non — 1 ! La raison en est que les bases théoriques mathématiques qui justifient le traitement de la logique au moyen d'opérations codifiées ont été jetées par George Boole vers 1848, en analogie consciente et voulue avec le traitement des nombres en arithmétique ou en algèbre. Le calcul logique qui en résulte est pour cela baptisé par l'usage *algèbre de Boole*.

Si nous combinons les opérations que nous venons de définir, nous créons des expressions logiques complexes, dont le traitement, la résolution ou la simplification sont assez proches de leurs équivalents en algèbre classique :
x.y + y.z + u.w + (a + b).(a.b + c.d) est ainsi une expression logique (ou *booléenne*, prononcer «bouléienne» pour paraître initié) susceptible d'être analysée algébriquement, ce que nous ne ferons pas, mais aussi d'être «cablée» concrètement avec des portes à base de transistors.

C'est sur cet aspect concret, bien sûr, que nous insisterons. Les paragraphes suivants vous montreront l'emploi des portes logiques dans quelques questions du genre de celles qui ne manqueront pas de se poser lors de la construction de Nanostar.

2 - 2 - 3. Quelques exemples

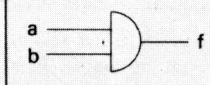
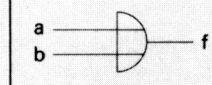
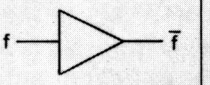
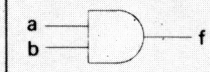
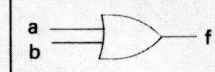
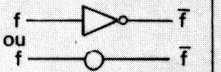
a) Le problème du comparateur de signes

Il s'agit de créer un circuit qui envoie un type de signal lorsque les signes de deux nombres sont identiques (tous deux + ou tous deux —) et un autre type lorsque les signes sont différents. Ce genre de comparaison est utile pour effectuer des multiplications : souvenons-nous des règles + par + égale +, — par + égale —, — par — égale +.

Introduisons quelques conventions arbitraires : un signe + sera représenté par 1, un signe — par 0 et la sortie vaudra 1 si les signes sont semblables, 0 si les signes sont différents.

Dans un ordinateur classique, la situation est fréquemment de ce type : un bit dans la représentation d'un nombre, appelé «bit de signe», représente le signe du nombre selon notre convention (ou la convention inverse, plus fréquente, ce qui est immatériel pour notre démonstration). Ce sont ces deux bits de signe que notre circuit doit recevoir.

Appelons-les A et B, et appelons S la sortie désirée. Avec nos conventions nous pouvons

forme	porte ET	porte OU	porte NON	emplois
1.				Schémas théoriques
2.				Schémas électroniques de principe
3.	S = f = a.b	S = f = a + b	$S = \overline{f}$	notation en algèbre de Boole
4.	f = a ∧ b	f = a ∨ b	~ f ou ¬ f	notation en logique
5.	f = a ∩ b	f = a ∪ b	$f' = \overline{f}$	notation en mathématique des ensembles
6.	f égale a et b	f égale a ou b	f barre ou non — f	prononciation de 3.

représentations schématiques des portes logiques

dresser la table de vérité du comparateur de signes :

comparateur de signes	A	B	S
	0	0	1
	0	1	0
	1	0	0
	1	1	1

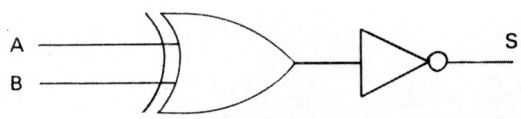
En comparant aux portes OU, OUX, ET, NON, décrites plus haut, nous voyons qu'aucune ne possède la même table : il nous faudra donc innover.

Remarquons que le OUX, ou OU exclusif, possède une table de vérité qui est en quelque sorte le «négatif» de celle de notre comparateur :

OUX	A	B	$A \oplus B$
	0	0	0
	0	1	1
	1	0	1
	1	1	0

S cherché	$\overline{(A \oplus B)}$
1	1
0	0
0	0
1	1

Si nous changeons en 1 les 0 de A + B et vice-versa, nous trouvons bien les valeurs de S que nous cherchons. Or, changer les 1 en 0 et les 0 en 1, c'est passer par la porte NON ! Voici la solution : faire un OU exclusif sur A et B, puis un NON sur le résultat. Algébriquement, on écrira :
$S = \overline{(A + B)}$, la barre portant sur toute l'opération. Electriquement, on dessinera le schéma suivant :



Malheureusement, la porte OUX est rarement employée. Essayons donc de nous en passer, malgré sa commodité sur notre exemple.

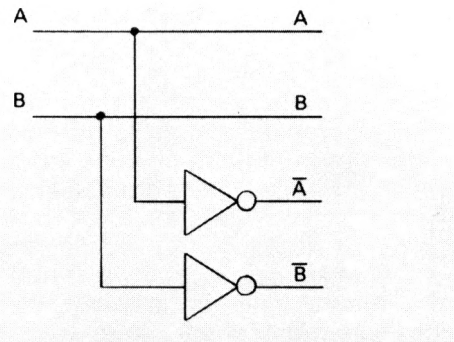
Remarquons que la table de vérité du comparateur ressemble à celle de ET, à l'exception du cas A = 0, B = 0 qui donne S = 1 et non 0 comme dans ET. Mais dans ce cas $\overline{A} = 1$ et $\overline{B} = 1$. Essayons donc *deux* ET : l'un sur A.B, l'autre sur $\overline{A}.\overline{B}$:

S cherché	A	B	A.B	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$	$A.B + \overline{A}.\overline{B}$
1	0	0	0	1	1	1	1
0	0	1	0	1	0	0	0
0	1	0	0	0	1	0	0
1	1	1	1	0	0	0	1

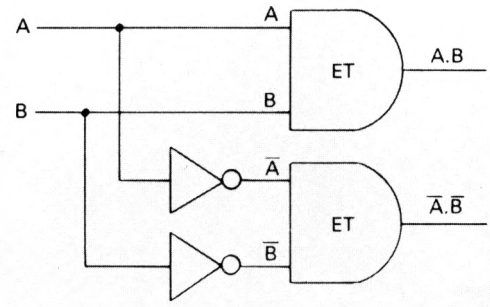
Relions A.B et $\overline{A}.\overline{B}$ par un OU et comparons avec S : nous trouvons la même forme. Ecrivons alors $S = A.B + \overline{A}.\overline{B}$

Il faut maintenant réaliser le circuit électronique correspondant.

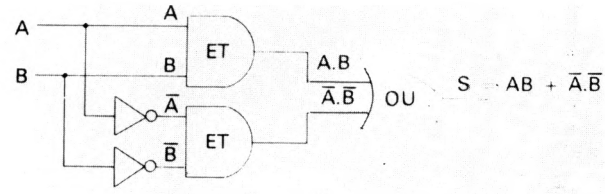
Réalisons d'abord $\overline{A}$ et $\overline{B}$ au moyen de portes NON, tout en conservant A et B dont nous aurons besoin plus tard :



Puis construisons A.B et $\overline{A}.\overline{B}$ au moyen de portes ET :



Enfin effectuons un ou inclusif sur A.B et $\overline{A}.\overline{B}$ au moyen d'une porte OU :



Voici donc réalisé notre comparateur de deux manières différentes, la plus simple incontestablement reposant sur un circuit OUX. Les amateurs de ce genre de travaux noteront qu'il existe bien d'autres circuits possibles réalisant la même fonction. L'homme de l'art réduit le circuit à sa plus simple expression d'après des règles mathématiques ou empiriques. Et nous observons qu'à la plus grande simplicité d'expression correspondra le coût de réalisation le plus faible, puisque chaque porte ET, OU, NON employée, il nous faudra l'acheter. L'élégance mathématique réduit les prix de revient : belle leçon !

b) Le décodeur binaire-décimal

Nous avons vu plus haut que l'ordinateur travaille sur des représentations *binaires* des nombres ou de la logique. Lorsqu'il s'agit de nombres, il est préférable pourtant que le résultat d'un calcul soit affiché dans notre bon vieux système *décimal.*

A cette fin, nous devrons souvent disposer d'un «décodeur» traduisant automatiquement le binaire en décimal. En voici le principe.

Le tableau suivant donne l'équivalent binaire de chacun des dix chiffres décimaux 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Rappelons simplement que le principe mathématique de la représentation d'un nombre dans le système à base 2 consiste à décomposer ce nombre en tas successifs de 8, 4, 2 et 1, lesquels sont les «puissances» de 2 : $2_1 = 2$, $2_2 = 2 \times 2 = 4$, $2_3 = 2 \times 2 \times 2 = 8$. Par convention, on décrète : $2_0 = 1$. Ainsi $7 = 4 + 2 + 1$ s'écrira 111, la position de la droite vers la gauche, de chaque 1 ou 0 donnant le nombre de puissances correspondantes de 2 utilisées ; 5 s'écrit 101 car il contient un tas de puissance 2 (4) zéro tas de puissance 1 (2) et un tas de puissance 0 (1).

Pour présenter un chiffre décimal, quatre chiffres binaires, quatre *bits* sont donc nécessaires. Ces 4 bits seront *l'entrée* de notre décodeur. La *sortie* sera constituée par dix fils, marqués de 0 à 9, et nous espérons recueillir une impulsion seulement sur le bon fil, entendons par là le fil dont le numéro correspond bien à la valeur décimale du binaire entré. Ce binaire entre par 4 fils, marqués A_0, A_1, A_2, et A_3 sur le tableau, l'indice correspondant à la puissance de 2 qu'enregistre le fil.

Ainsi, un signal 1 sur l'entrée A_3 correspond à un tas 8. Mais pour savoir s'il s'agit de 8 ou de 9, il nous faut vérifier A_0 : si le signal reçu par A_0 est 0, c'est 8, sinon c'est 9. En fait, pour décoder sans ambiguïté, nous devrons vérifier l'état de chacun des bits d'entrée A. Nous le ferons en câblant son entrée non seulement le fil A_i, mais aussi la valeur $\overline{A_i}$, au moyen de circuits NON. En sortie, des circuits ET vérifieront que la combinaison reçue est conforme. Le schéma de la *matrice de décodage* est représenté en entier.

tableau de conversion décimal-binaire

décimal	binaire			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
	A_3	A_2	A_1	A_0

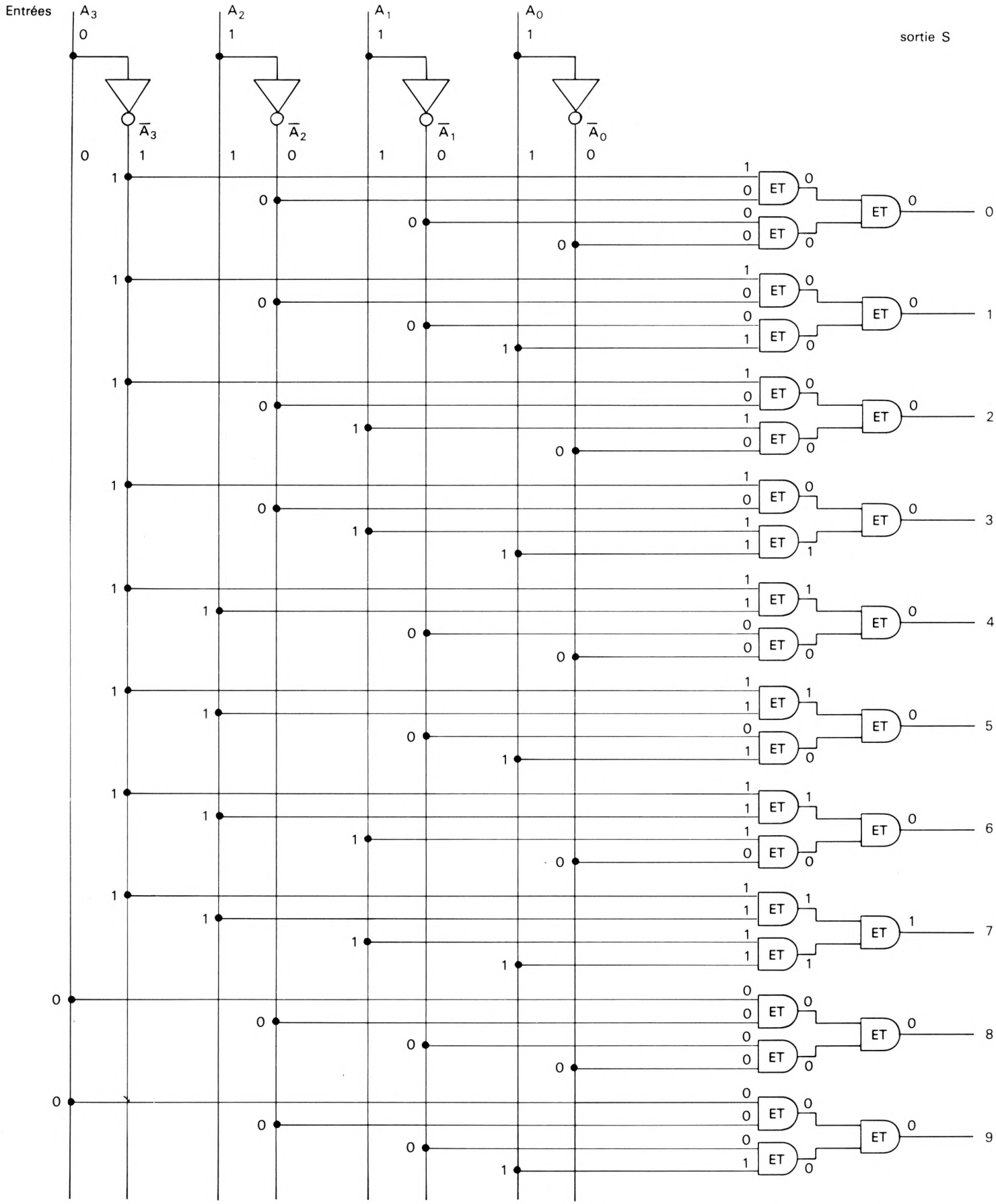
Ce type de matrice de décodage — qui, sur notre exemple, suppose que le nombre en sortie est représenté en «décimal codé binaire» et non en «binaire pur» — est d'emploi fréquent : nous en retrouverons un exemple avec le circuit de reconnaissance de la touche frappée au clavier, ou avec le circuit d'affichage lumineux de Nanostar. Toutes les matrices de décodage emploient de même des portes NON et ET.

2 - 2 - 4. Les portes négatives de la logique positive : NOR et NAND

Les conventions que nous avons adoptées jusqu'ici — 1 pour oui et 0 pour non — sont purement arbitraires. Pis, elles sont doublement arbitraires puisque le 1 a été associé au passage du courant (ou à une tension «forte») et le 0 à l'absence de courant (ou à une tension «faible»). Les électroniciens parlent dans ce cas de *logique positive.*

Malheureusement dans la pratique, comme nous le verrons, ils réalisent l'inverse, une *logique négative* : ainsi le circuit OU qu'ils fabriquent, au lieu de sortir une tension «forte» dans tous les cas *sauf* lorsque les deux entrées sont des tensions «faibles», ne sort une tension «forte» *que si* les deux entrées sont «faibles».

Pour conserver nos conventions, il faut donc admettre que ce circuit n'est pas un OU mais un OU suivi d'un NON : le résultat s'appelle *NOR* en anglais, et parfois *NI* en français (mais le terme français, peu employé, est ambigu : nous utiliserons donc l'appellation anglaise).



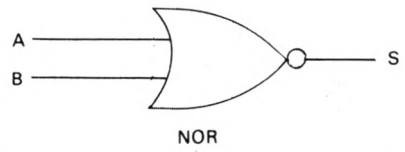
Matrice de décodage binaire → décimal. On y a figuré les différents états 0 et 1 conduisant au décodage 0111 → 7

La table de vérité de la porte NOR est la suivante :

NOR	A	B	S
	0	0	1
	0	1	0
	1	0	0
	1	1	0

Puisqu'elle est équivalente à un OU suivi d'un NON, on peut écrire en algèbre de Boole : $S = \overline{(A + B)}$, la barre du NON couvrant toute l'opération A + B.

Le symbole correspondant dans les schémas électroniques est le symbole du OU, auquel on a adjoint le petit cercle du NON : son mode de fonctionnement est ainsi parfaitement explicité.

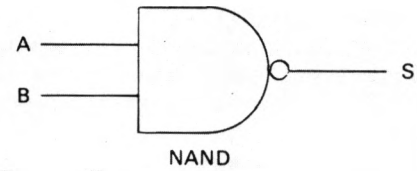


De façon analogue, la porte ET que l'on trouve dans le catalogue des fabricants de composants est fabriquée selon les principes de la logique négative : elle ne sort un courant «faible» que si les entrées sont fortes ; c'est, dans la logique positive à laquelle nous nous tenons, l'équivalent d'une porte ET suivie d'une porte NON. Nous l'appellerons *NAND*, à l'anglaise.

Sa table de vérité est la suivante :

NAND	A	B	S
	0	0	1
	0	1	1
	1	0	1
	1	1	0

En algèbre de Boole, nous écrirons $S = \overline{(A.B)}$, la barre couvrant l'ensemble de l'opération A.B, et le symbole électronique reprendra le symbole du ET suivi du rond du NON :



Avant de passer à l'utilisation pratique de ces portes, remarquons qu'en changeant les 0 en 1, et les 1 en 0 dans la table de vérité du NOR, c'est-à-dire en écrivant le NOR en logique négative, on obtient la table de vérité du NAND en logique positive, et vice-versa.

NOR logique positive	A	B	S
	0	0	1
	0	1	0
	1	0	0
	1	1	0

identique

NAND logique négative	A	B	S
	1	1	0
	1	0	0
	0	1	0
	0	0	1

NOR logique négative	A	B	S
	1	1	0
	1	0	1
	0	1	1
	0	0	1

identique

NAND logique positive	A	B	S
	0	0	1
	0	1	1
	1	0	1
	1	1	0

On constate donc que NAND et NOR sont deux aspects «négatifs» l'un de l'autre d'un même circuit (notons ici la similitude remarquable des expressions photographiques et électroniques, sûrement un des concepts à l'origine du stéréo-neutrinographe de Super Zap !). Cette remarque explique l'emploi pratique généralisé des NOR et NAND au lieu des formes «positives» OU et ET dans les schémas de construction. Mais rassurez-vous : dans tout ce qui suit, sauf avis contraire, les conventions seront celles de la logique positive : 1 pour tension «forte», O pour tension «faible».

2 - 2 - 5. Les portes de la création (17)

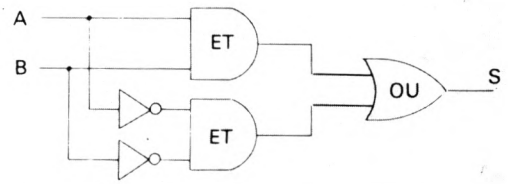
Le problème de la conception d'un circuit logique se pose à présent en ces termes : nous disposons d'une part de portes logiques au fonctionnement assez simple : NON, ET, OU que nous dessinerons dans les schémas de principe, d'au-

(17) Avec toutes nos excuses à Philip José Farmer, auteur d'un merveilleux roman du même titre, dans lequel le signal d'ouverture d'une porte n'est pas un courant électrique mais une onde sonore émise par un instrument de musique fabuleux. (Collection Galaxie bis, Editions Opta, Paris 1971).

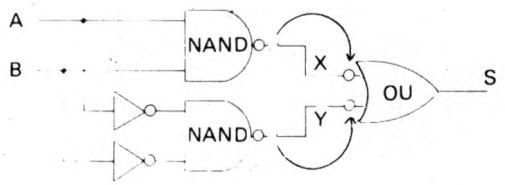
tre part de portes NAND et NOR, moins simples assurément mais qui, avec NON, sont les seules utilisables dans la pratique : ce sont ces dernières que nous utiliserons pour construire Nanostar, et non ET et OU.

Fort heureusement, une technique assez simple permet de passer du schéma de principe au schéma de réalisation concret. Prenons un exemple, d'où nous tirerons la méthode générale : le comparateur de signes étudié plus haut.

La formule d'algèbre de Boole à réaliser est $S = A.B + \overline{A}.\overline{B}$. Le circuit théorique qui s'en déduit, à base de NON, ET et OU est le suivant :



Pour le transformer en un circuit identique du point de vue logique, mais à base de NOR et NAND, commençons par transformer les ET à gauche en NAND : il suffit de leur ajouter un cercle de négation. Mais pour ne pas modifier la valeur logique du courant il faut ajouter à chaque fois *deux* cercles : une double négation est équivalente à une affirmation, $\overline{\overline{A}} = A$. Nous propageons ensuite le deuxième cercle en le faisant glisser le long du fil, vers la droite, comme une boule sur la tringle d'un boulier, jusqu'à ce qu'il rencontre une porte. Le résultat ressemble au dessin que voici :



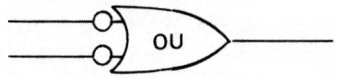
Nous voici donc avec deux NAND, circuits autorisés, et un OU non autorisé mais dont les deux entrées sont précédées par une négation.

Appelons X et Y les valeurs logiques de ce qui sort des NAND. Le OU précédé de deux négations nous sort $S = \overline{X} + \overline{Y}$.

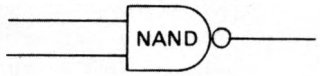
Dressons la table de vérité de $\overline{X} + \overline{Y}$:

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X} + \overline{Y}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

Comparons le résultat avec la table de vérité du circuit NAND : nous observons que c'est la même, ce qui nous réjouit puisque NAND est un circuit autorisé dans la réalisation pratique. Nous sommes d'autant plus satisfaits que la loi trouvée est générale : chaque fois que nous rencontrerons

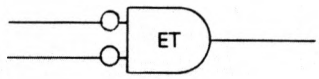


nous pourrons le remplacer par

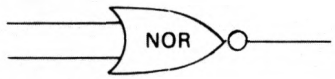


ce qui revient, remarquons-le, à écrire $\overline{X} + \overline{Y} = \overline{(X.Y)}$. Les spécialistes auront reconnu l'une des célèbres lois d'Augustus de Morgan.

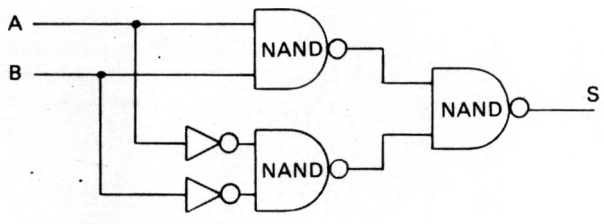
L'autre loi de de Morgan s'écrit $\overline{X}.\overline{Y} = \overline{(X + Y)}$; elle est facile à vérifier au moyen des tables de vérité, et nous conduit à remplacer



par



Revenons à notre comparateur de signes : son schéma de réalisation est devenu :



il n'y subsiste plus que des NON et des NAND. Nous pouvons donc le construire.

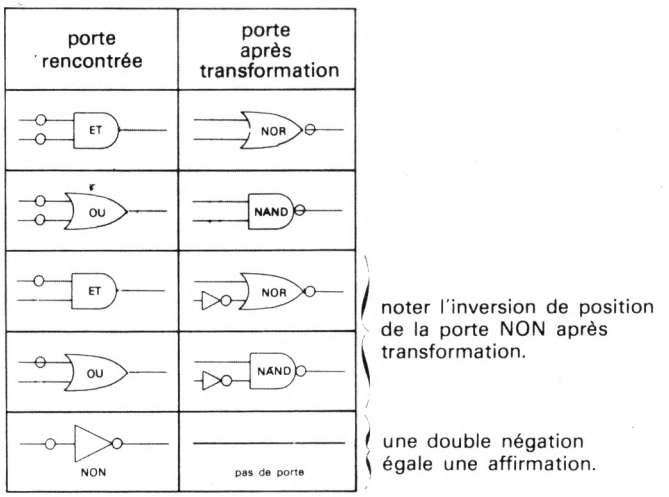
De manière générale, la méthode de conception d'un circuit logique complexe comporte quatre phases :

1. Dresser la table de vérité du circuit à dessiner, ou, ce qui revient au même, écrire

l'expression booléenne de la sortie désirée en fonction des entrées.

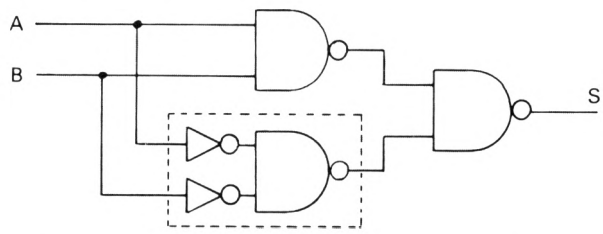
2. Dessiner le schéma de principe au moyen des portes simples NON, ET, OU. (Exclure autant que l'on peut le OU exclusif).

3. En partant de la gauche du schéma de principe, transformer les ET en NAND et les OU en NOR en ajoutant deux négations à chaque fois. Propager vers la droite les négations excédentaires et transformer les portes rencontrées selon le tableau ci-dessous :

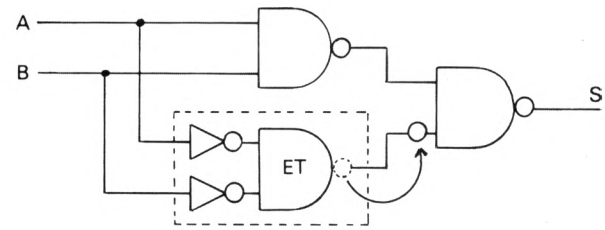


4. Vérifier que le résultat ne comporte plus que des NON, des NOR et des NAND, essayer de simplifier si cela apparaît possible, et transmettre à la fabrication après avoir vérifié par une table de vérité que le circuit dessiné remplit bien les fonctions désirées.

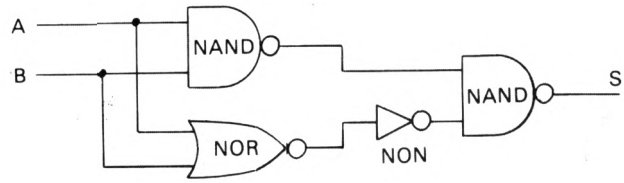
Les essais de simplification sont intéressants dans la mesure où, réduisant le nombre de portes, ils réduisent le prix de revient de l'ensemble. Par exemple notre comparateur de signes comprend 5 portes : 2 NON et 3 NAND. Il est possible de le ramener à 4 portes selon le processus suivant ; voici le comparateur avant simplification :



Observons que nous avons un NAND précédé de 2 NON dans le rectangle pointillé. Faisons glisser le cercle du NAND jusqu'à la porte suivante :



il nous reste un ET (NAND ayant perdu la boule !) précédé de deux NON, c'est-à-dire un NOR (voir tableau ci-dessus). La boule devant le NAND qui précède immédiatement la sortie se traduit par un NON, et le circuit prend la forme optimisée suivante :



il n'y a plus que quatre portes : 2 NAND, 1 NOR et 1 NON au lieu de cinq : nous avons économisé 20 % du prix du comparateur en nous livrant à ce petit exercice. Les amateurs d'algèbre de Boole vérifieront que ce circuit donne

$S = [\overline{(\overline{A}.B)}.\overline{(A + \overline{B})}]$ qui est bien égal à $S = A.B + \overline{A}.\overline{B}$.

La simplification de circuits plus complexes est une technique délicate et très empirique. L'exemple du comparateur nous a montré que l'expression algébrique simple ($S = A.B + \overline{A}.\overline{B}$) n'est pas obligatoirement la forme pratique la plus simple à réaliser. De nombreuses méthodes ont été imaginées, mais elle varient selon le but recherché et nous n'en dirons pas plus.

Appliquons notre méthode de conception à un circuit réel que nous aurons à concevoir lors de la réalisation de Nanostar.

2 - 2 - 6. Un exemple de logique combinatoire : l'autorisation d'accès à la mémoire

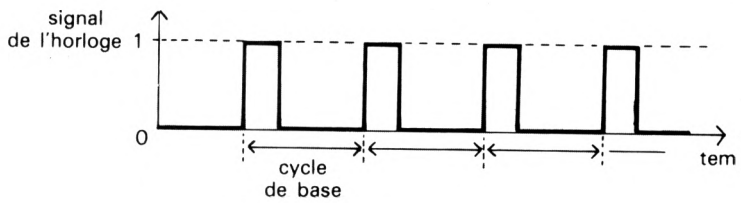
Comme tout bon ordinateur, Nanostar est doté d'une unité qui réalise des fonctions de calcul et de contrôle : ce sera le microprocesseur, et d'une mémoire centrale où sont stockées les données en cours de traitement. Lorsque nous voudrons lire ou écrire dans la mémoire centrale, il nous faudra y accéder. Pour cela, nous ferons appel à une *autorisation d'accès à la mémoire*. En effet, pour y

accéder, certaines conditions doivent être remplies :

a) Il faut avoir une raison valable. On ne rentre pas dans la mémoire centrale comme dans un moulin, elle contient quelques trésors à protéger farouchement. Deux raisons sont considérées comme valables :
— *la lecture* : nous voulons extraire une information qui y est stockée pour l'envoyer vers le processeur ;
— *l'écriture* : nous voulons stocker dans la mémoire centrale une information qui provient d'ailleurs.

b) Pour procéder à la lecture, il faudra tout d'abord que le processeur la demande : il émettra, à cette fin, un signal que nous appellerons MEM (comme mémoire) qui vaudra 1 lorsque la demande est formulée, 0 dans les autres cas.

D'autre part, on n'accède pas à la mémoire n'importe quand : le rythme de travail de l'unité centrale est conditionné par son cycle de base. Ce cycle est «battu», au sens où le métronome «bat» la mesure, par une horloge qui émet un signal 1 à intervalles réguliers :



Enfin un signal dit «de synchronisation», régulier, permet à l'horloge de «passer» dans le circuit (cf. plus haut).

Le signal de l'horloge sera appelé Φ_1, nous aurons donc $\Phi_1 = 1$ au début du cycle, $\Phi_1 = 0$ ensuite. (Il s'agit de Phi, la lettre grecque. Le 1 distingue Φ_1 d'un Φ_2 que nous verrons plus tard.

Le signal de synchronisation sera appelé SYNC.

Pour qu'une autorisation de lecture soit envoyée, il faudra que MEM, Φ_1 et SYNC vaillent 1 en même temps. Cela se traduira par l'expression booléenne MEM. (Φ_1. SYNC) qui ne vaut 1 que si chacun des termes vaut 1.

c) Pour que *l'écriture* soit possible, il faut qu'aucune opération d'écriture à partir de la mémoire ne soit en cours.

A une opération d'écriture en cours correspond un signal que nous écrirons WO (pour Write Output, Ecriture en Sortie).

A une opération d'écriture vers un périphérique de sortie correspond un signal que nous appellerons OUT (comme Output, Sortie).

Lorsque ni WO ni OUT ne valent 1, $\overline{WO}.\overline{OUT}$ vaut 1, et seulement dans ce cas.

De plus, pour autoriser l'écriture, il faut synchroniser les opérations comme pour la lecture. Nous devons donc avoir simultanément Φ_1. SYNC = 1 et $\overline{WO}.\overline{OUT} = 1$, ce qui impose que le produit ($\overline{WO}.\overline{OUT}$) ($\Phi_1$. SYNC) vaille 1.

d) Récapitulons : pour que le signal d'autorisation d'accès à la mémoire soit en fonction, il faut *soit* que les conditions de demande de lecture, *soit* les conditions de demande d'écriture soient remplies. Nous relions donc conditions de lecture et conditions d'écriture par un OU pour obtenir notre signal d'accès :

$$S = \underbrace{MEM.(\Phi_1.SYNC)}_{\text{lecture}} + \underbrace{(\overline{WO}.\overline{OUT})(\Phi_1.SYNC)}_{\text{écriture}}$$

Une mise en facteur simple, comme dans l'algèbre traditionnelle, nous permet d'écrire :

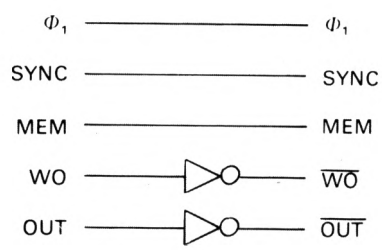
$S = (\Phi_1.SYNC).(MEM + \overline{WO}.\overline{OUT})$.

La *première phase* de notre méthodologie de conception de circuits est terminée ; elle a consisté essentiellement à identifier les facteurs qui entraînent le déclenchement d'une action : les informaticiens de gestion noteront que nous avons dressé la «table de décision» du problème, puis utilisé le symbolisme algébrique pour la résumer.

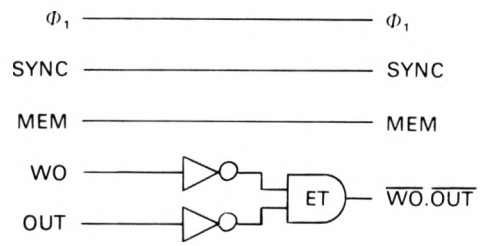
Passons à la *deuxième phase* : représentation du circuit avec des portes simples NON, ET, OU.

Nous disposons de cinq entrées : Φ_1, SYNC, MEM, WO et OUT.

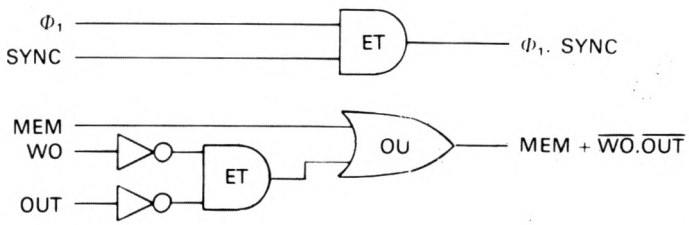
Observons d'abord que WO et OUT n'apparaissent que sous leur forme négative, et dessinons le schéma suivant :



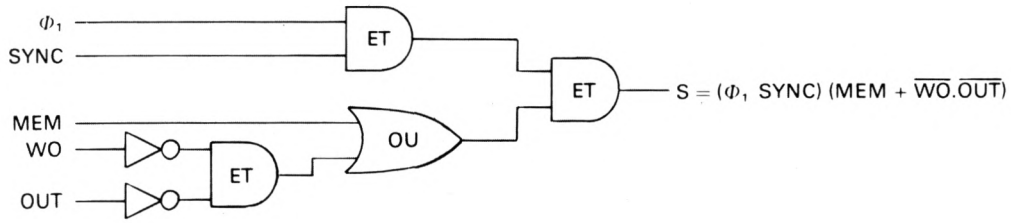
Puis réalisons le produit logique $\overline{WO}.\overline{OUT}$:



Effectuons le produit Φ_1. SYNC et la somme MEM + $\overline{WO}.\overline{OUT}$:

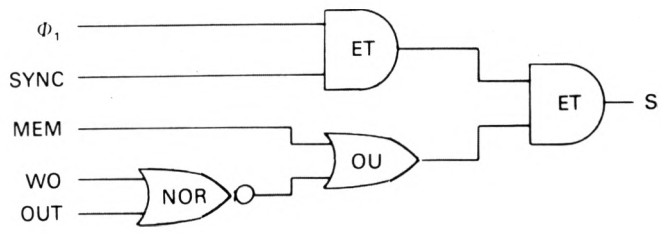


Enfin réalisons le produit des deux expressions intermédiaires :

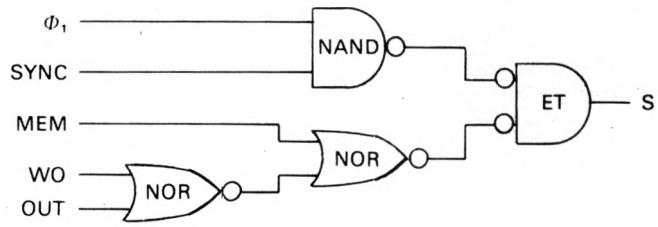


Voilà donc le schéma de principe de la commande d'accès à la mémoire centrale.

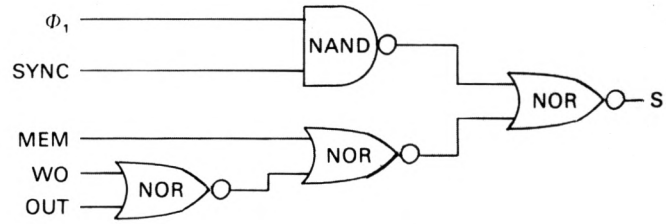
La *troisième phase* : dessin du schéma pratique avec NON, NOR et NAND, commence par l'élimination des NON qui précèdent le ET ; ils le transforment en NOR.



Transformons le ET et le OU du deuxième étage en NAND et NOR respectivement :



Les deux NON engendrés devant le ET du 3e étage le transforment en NOR, et le schéma pratique est réalisé :



Vérification avec quelques valeurs des variables :

a) Φ_1 = 1, SYNC = 1, MEM = 1, WO = 1, OUT = 1 :
Φ_1 = 1 et SYNC = 1 donnent 0 avec NAND.
WO = 1 et OUT = 1 donnent 0 avec NOR, le NOR avec MEM = 1 donnant 0.
Le dernier NOR avec 0 et 0 en entrée donne S = 1, l'accès est autorisé (pour la lecture seulement).

b) Φ_1 = 0, les autres variables valant 1 :
Φ_1 = 0, et SYNC = 1 donnent 1 avec NAND.
Le dernier NOR avec 0 et 1 en entrée donne S = 0, l'accès est interdit par manque de synchronisation.

c) Φ_1 = 1, SYNC = 1, MEM = 0, WO = 1, OUT = 0 : Φ_1 et SYNC donnent 0 avec NAND.
WO = 1 et OUT = 0 donnent 0 avec le premier NOR, le deuxième donne 1 avec MEM = 0.

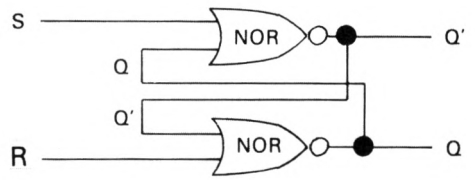
Le troisième étage avec 1 et 0 donne S = 0, l'accès est interdit puisqu'aucune demande de lecture n'est formulée et qu'une opération d'écriture est en cours (WO = 1).

Les problèmes logiques que nous avons résolus jusqu'ici appartiennent à une classe que les électroniciens appellent *logique combinatoire*. En effet, la valeur d'une variable de sortie y dépend uniquement de la valeur des variables d'entrée, par une *combinaison* de portes logiques qui s'ouvrent et se ferment selon les signaux reçus. Mais ce qui s'est passé auparavant n'influe aucunement sur l'état de la sortie : la logique combinatoire n'a pas de mémoire et oublie tout dès qu'elle a accompli sa tâche.

A l'opposé, ou plutôt dans un autre domaine, nous trouverons d'autres circuits, astucieux montages des portes que nous venons d'étudier, qui passent par plusieurs états de transition avant de se stabiliser dans une position parmi un petit nombre qui, seules, sont stables : c'est le domaine de la logique dépendant du temps. L'état d'un signal y dépend de l'état de ce signal à un instant antérieur. On aurait pu l'appeler «logique temporelle» l'usage a préféré *logique séquentielle*.

2 - 2 - 7. Un jeu de bascule électronique : les flip-flops et la logique séquentielle

Amusons-nous à réaliser le montage suivant, qui paraît assez peu orthodoxe au premier coup d'œil :



La sortie de chaque porte NOR est réinjectée en entrée de sa voisine.

Essayons de voir ce que donne le système. Pour cela, comme les sorties Q' et Q influent sur les entrées, faisons des hypothèses à leur sujet :

Imaginons qu'à un instant t donné R vaut 0, S vaut 0 et Q_t vaut 0.

Dressons progressivement la table de vérité du système à l'instant suivant t + 1.

R	S	Q_t	Q'_t	Q_{t+1}	Q'_{t+1}
0	0	0	1	0	1

Le NOR qui a Q et S comme entrées sont Q', qui ne peut donc valoir que 1, puisque Q + S = 0 et $Q' = (\overline{Q + S}) = 1$. De manière générale, on voit que Q' ne peut valoir que $\overline{Q}$.

Le NOR qui a Q' et R pour entrées a donc 1 et 0 en entrée, il sort donc 0 à l'instant t + 1.Q'_{t+1} reste égal à $\overline{Q_{t+1}} = 1$.

Observons donc que le système reste dans l'état où il était : Q = 0. $\overline{Q} = 1$.

Essayons $Q_t = 1$; $Q'_t = \overline{Q_t} = 0$; le NOR qui a Q' et R en entrée sort Q = 1, le système est encore stable :

R	S	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1

Rien ne peut donc nous indiquer l'état réellement pris par le système puisque les sorties Q = 0 et Q = 1 sont également possibles.

Passons donc à R = 1, S = 0 ; essayons $Q_t = 0$, $\overline{Q_t} = 1$; le NOR avec S = 0 et Q + 0 sort $\overline{Q} = 1$ et le NOR avec $\overline{Q} = 1$, et R = 1 et R = 1 sort Q = 0. Le système est encore stable :

R	S	Q_t	Q_{t+1}
1	0	0	0

Testons, toujours avec R = 1, S = 0, le cas où $Q_t = 1$, $\overline{Q} = 0$; le NOR avec S = 0 et $Q_t = 1$ sort Q = 1. Avec R = 1 et $\overline{Q} = 1$, le second NOR sort $Q_{t+1} = 0$. Le système est revenu dans l'état Q = 0, d'où il ne sortira plus tant que quelqu'un n'aura pas modifié R ou S !

R	S	Q_t	Q_{t+1}
1	0	0	0
1	0	1	0

On dit alors que le système a été réinitialisé (*Reset* en anglais, d'où la lettre R). Si R repasse à 0, Q restera indéfiniment à 0 : il suffit d'envoyer un bref signal 1 sur R pour remettre notre circuit à 0.

On voit de façon tout à fait similaire qu'une valeur 1 pour S positionne Q_{t+1} à 1 quelle qu'ait été la valeur de Q_t (si R est à 0 !).

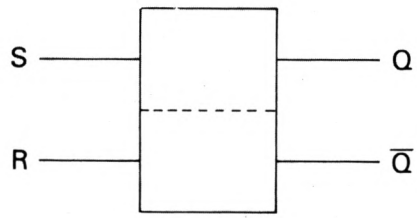
R	S	Q_t	Q_{t+1}
0	1	0	1
0	1	1	1

On dit alors que le système est positionné à 1 (*Set* en anglais, d'où la lettre S). S peut repasser à 0, Q restera indéfiniment à 1.

Observons que si dans la situation S = R = 0, Q vaut 0, nous pouvons affirmer que c'est R qui a reçu le dernier un signal 1. Si Q vaut 1, c'est S. Notre montage «se souvient» du dernier événement qui s'est produit en entrée. C'est une mémoire. Rudimentaire, certes, mais c'est tout de même une mémoire. Nous avons construit une mémoire avec des portes logiques, ce qui est assez inattendu.

Que se passe-t-il si R et S valent simultanément 1 ? Le système devrait sortir soit simultanément Q = Q' = 0, soit Q = Q' = 1 ce qui est impossible. La situation ne doit pas se produire et l'on évitera de faire en même temps R = S = 1, le résultat étant imprévisible. Voilà un cas qui évoque les antiques paradoxes de la logique tels que «ce que je suis en train d'écrire est faux» ; la phrase et son contraire doivent être vraies (ou fausses) simultanément, ce qui renie Aristote. Préservons Nanostar de ces hérésies...

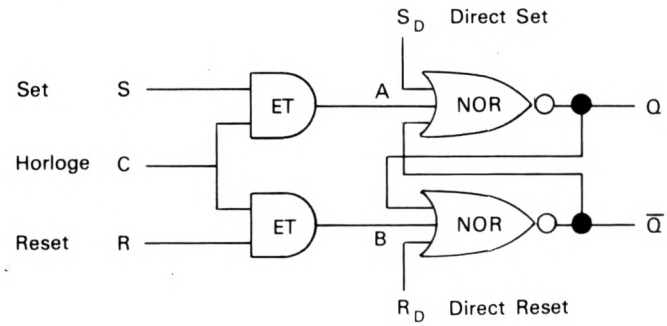
L'objet que nous venons de concevoir porte plusieurs noms : à l'anglaise, très imagé, nous l'appellerons *flip-flop*, ce qui évoque bien ces deux positions stables («flip» : jeter, sauter, basculer, est associé à un certain mouvement vers le haut, «flop» : retomber, s'avachir, être mou, à un mouvement vers le bas). Plus sérieusement, à la française, nous le désignerons par *bascule* ou plus rarement *bistable*, et comme nous verrons plusieurs variantes de ce même schéma de base, nous spécifierons *bascule R/S* et utiliserons le symbole suivant :



en en rappelant la table de vérité :

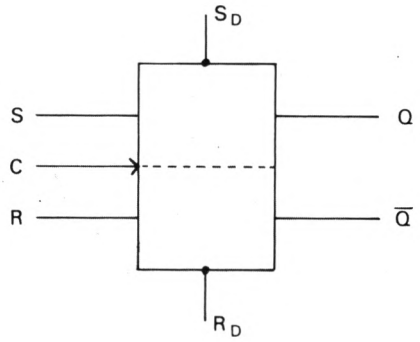
bascule R/S	R	S	Q_t	Q_{t+1}
	0	0	0	0
	0	0	1	1
	0	1	0	1
	0	1	1	1
	1	0	0	0
	1	0	1	0
	1	1	–	?

Le temps intervient dans la logique séquentielle : afin de le contrôler, les bascules répondent souvent uniquement à certains instants fixés par une horloge émettant un signal 1 à intervalles réguliers. Le flip-flop correspondant est désigné sous le nom de *bascule C*, C comme «clock», l'horloge en anglais. Son schéma est le suivant :

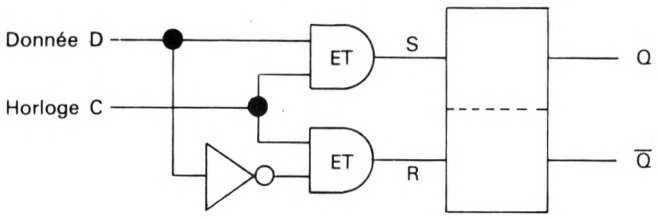


On voit que A et B valent 0 tant que C, l'horloge, vaut 0. Lorsque C vaut 1, A et B valent S et R respectivement et la bascule C fonctionne alors comme une bascule R/S classique. Ce montage permet de synchroniser la bascule au rythme de l'horloge. Il est souvent complété par des entrées S_D et R_D, Direct Set et Direct Reset qui autorisent des mises à 1 ou remises à 0 directes, sans passer par l'horloge.

Le symbole de la bascule C est le suivant :



Un autre type de bascule est employé pour mémoriser une donnée binaire de façon simple : une seule entrée D est utilisée en plus de l'horloge C. Lorsque l'horloge vaut 1, la sortie Q est identique à l'entrée D. Dès que l'horloge passe à 0 (on dit alors dans le jargon technique qu'elle «tombe»), la sortie Q conserve la dernière valeur prise par D avant la chute de l'horloge. Le montage qui permet de réaliser cette bascule est le suivant :

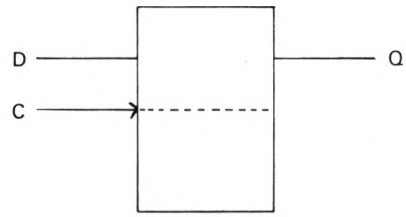


L'introduction du NON, c'est-à-dire de D dans le deuxième ET permet de surcroît d'éviter le cas litigeux S = R = 1 puisque S = 1, R vaut obligatoirement 0.

La table de vérité de cette bascule, appelée *bascule type D* (le D veut dire Data ou Donnée) est la suivante :

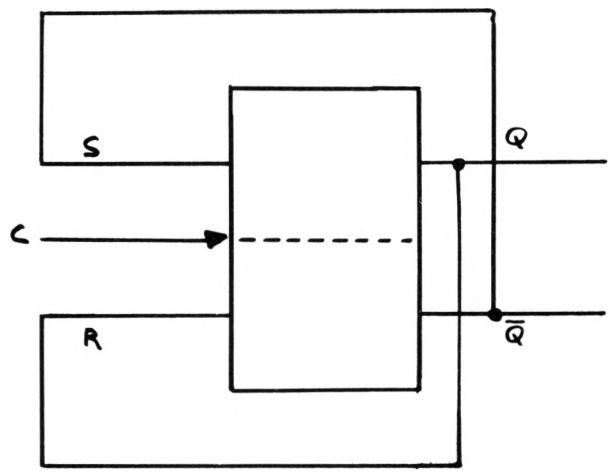
bascule D	D	C	Q_t	Q_{t+1}
	0	0	0	0
	0	0	1	1
	0	1	0	0
	0	1	1	0
	1	0	0	0
	1	0	1	1
	1	1	0	1
	1	1	1	1

Le symbole de ce flip-flop type D est le suivant :



Une bascule type C, à l'horloge, peut être aisément transformée en «*compteur*». Un compteur est un bistable qui change d'état à chaque passage de l'horloge, c'est-à-dire à chaque arrivée d'un bit «1» sur le circuit de l'horloge.

Voici le schéma-type d'un compteur :

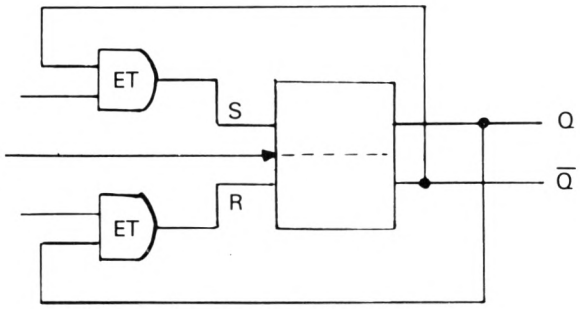


Si l'état initial est Q = 1, $\overline{Q}$ = 0, alors S = 0 et R = 1. A l'arrivée de l'horloge (C = 1), Q est positionné comme S, donc Q = 0 et $\overline{Q}$ = 1, et l'on passe à S = 1 et R = 0. Au passage suivant de l'horloge Q = 1 et $\overline{Q}$ = 0, et l'on revient à l'état initial, etc. Si tic vaut 0 et tac vaut 1, le compteur fait «tic-tac, tic-tac, tic-tac...» au rythme de l'horloge.

Notons que dans la pratique, il faut introduire un retard dans le circuit pour éviter les ambiguïtés signalées plus haut. C'est ce à quoi s'emploient diodes, résistances et capacités montés par le constructeur sur le compteur.

Un dernier type de bascule combine les possibilités de la bascule R-S et celles du compteur. C'est la bascule *J-K*, dite encore *J-K flip-flop*.

Le montage est le suivant :



Si J = 0 et K = 1 et Q_t = 1, lorsque l'horloge passe à 1, S donc Q_{t+1} passe à 0. La table de vérité du J-K flip-flop est :

J-K	J	K	Q_t	Q_{t+1}
	0	0	0	0
	0	0	1	1
	0	1	0	0
	0	1	1	0
	1	0	0	1
	1	0	1	1
	1	1	0	1
	1	1	1	0

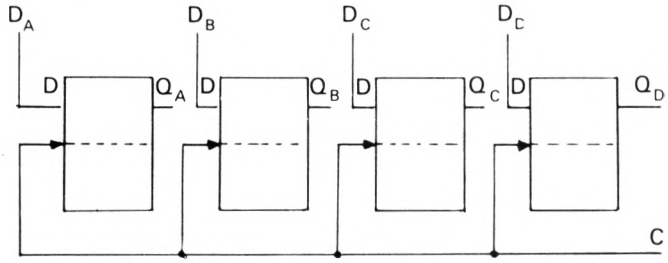
que l'on peut mettre sous la forme condensée suivante, qui rend mieux compte des phénomènes :

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q}_t$

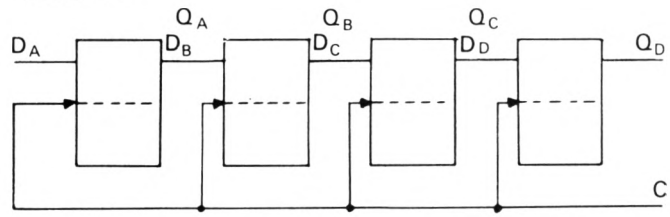
Lorsque J = K = 0, la sortie du flip-flop reste la même avant et après le passage de l'horloge. Si J = K = 1, la sortie est inversée après passage de l'horloge et le système se comporte tout à fait comme un compteur. Si J = K, au contraire, le système se comporte comme une bascule C classique, J jouant le rôle de R et K le rôle de S.

Comme dans le cas d'un compteur, un certain nombre de retards sont introduits dans le montage dont le schéma réel est légèrement plus compliqué que celui que nous avons donné.

La logique séquentielle que nous venons de décrire rapidement trouve un grand nombre d'applications dans les ordinateurs. Pour construire des «*registres*» par exemple, systèmes qui vont mémoriser une suite de chiffres binaires. Voici un registre de 4 positions binaires, composé de 4 bascules type D synchronisées sur la même horloge, et qui vont donc enregistrer les quatre chiffres binaires D_A, D_B, D_C, D_D dès que l'horloge passera



Les informaticiens de gestion, ou plus généralement les programmeurs d'ordinateurs classiques, connaissent bien les instructions de *décalage* qui dans le cas d'un décalage vers la droite par exemple (*shift right* en anglais), décalent une suite de bits de n positions vers la droite en comblant à gauche par des zéros (délaissons dans ces exemples simples les décalages circulaires qui font réapparaître à gauche les bits perdus à droite). Voici un registre qui à chaque passage de l'horloge décale tout à droite d'un cran ; il suffit de penser à relier les sorties Q_A, Q_B, Q_C aux entrées D_B, D_C, D_D respectivement sur le schéma précédent :

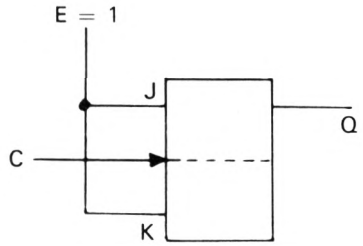


A chaque passage de l'horloge, après une situation initiale 1000, le registre contient :

horloge	Q_A	Q_B	Q_C	Q_D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	0	0	0	0

On voit progresser le 1 vers la droite jusqu'à ce qu'il disparaisse au 4e top !

Avec des J-K flip-flops, il est possible de créer des *compteurs à propagation (ripple-through counters* en anglais) qui permettent des «divisions» de la fréquence de l'horloge et nous mènent droit aux compteurs binaires : considérons pour commencer le montage suivant, à base de bascule J-K :

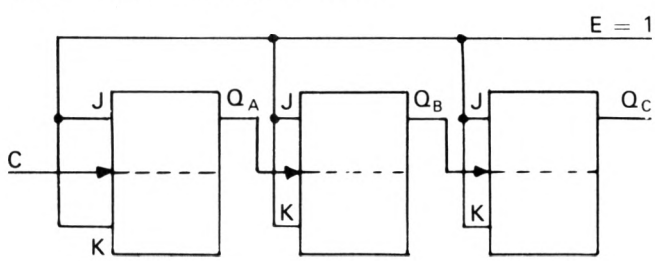


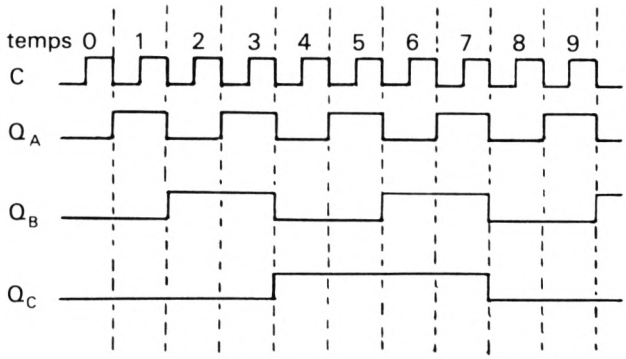
L'entrée E est mise en permanence à «1». D'après la table de vérité du J-K flip-flop, avec J = K = 1, $Q_{t+1} = \overline{Q_t}$. Supposons que l'horloge ait la forme classique suivante (elle passe régulièrement, à intervalles de temps égaux, par 1 et 0) :

La sortie Q, le lecteur le vérifiera facilement, prend alors la forme :

et l'on voit que Q a une fréquence moitié de la fréquence de C.

Mais l'on peut poursuivre l'opération en utilisant Q comme entrée d'un deuxième J-K positionné en permanence à 1. La sortie de ce deuxième J-K flip-flop aura une fréquence égale au quart de la fréquence initiale de C, puis un troisième sortira une fréquence égale à un huitième, etc. Avec trois J-K flip-flop, le montage suivant permet de réaliser un compteur binaire dont la valeur donne le nombre de passages d'horloge effectués depuis un instant initial :
la table des états Q_C, Q_B, Q_A montre bien le processus de comptage binaire :





Q_C	Q_B	Q_A	temps
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7
0	0	0	8
0	0	1	9

L'ensemble de ces fonctions réalisées au moyen de bascules est donc essentiel à la constitution des éléments logiques de l'ordinateur.

De portes en bascules, nous avons déjà bien cheminé dans le labyrinthe logique de la machine : la «techno-logique» voit sa seconde moitié progressivement défrichée. Et si nous explorions maintenant un peu la première, qui la concrétise... en silicium ?

2 - 3. ACTE DEUX : LES CIRCUITS INTEGRES BIPOLAIRES

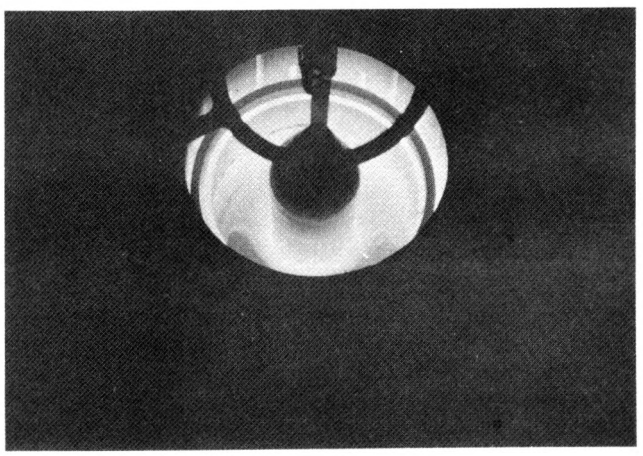
Le maître mot est lâché : *circuits intégrés*. Ce sont eux qui rendent possible aujourd'hui notre projet de micro-ordinateur, eux qui vont servir à matérialiser toute la logique de notre machine dont nous venons de présenter les constituants de base, eux enfin qui seront le microprocesseur, et même la mémoire centrale.

Physiquement, un circuit intégré se présente sous la forme d'un rectangle de silicium de dimensions millimétriques, épais d'une fraction de millimètre : la

microplaquette (en anglais *chip* ou *die*, et en jargon électronique familier, la *pastille* ou la *puce* !). Sur cette pastille, un nombre de composants électroniques (diodes, transistors, résistances, etc.) allant de quelques dizaines... à plusieurs milliers ! Par un procédé chimico-photo-lithographique extraordinairement ingénieux, que nous allons voir, on fait littéralement «pousser» les composants au sein même du silicium par toute une série d'opérations de grande précision, cuisine aux nombreux tours de main secrets, pour obtenir en fin de fournée simultanément plusieurs centaines de *circuits intégrés monolithiques* (ce dernier adjectif exprimant que la puce est en fait un «bloc» de silicium — si l'on peut dire vu sa taille — et non une juxtaposition de composants fabriqués indépendamment et rassemblés par la suite).

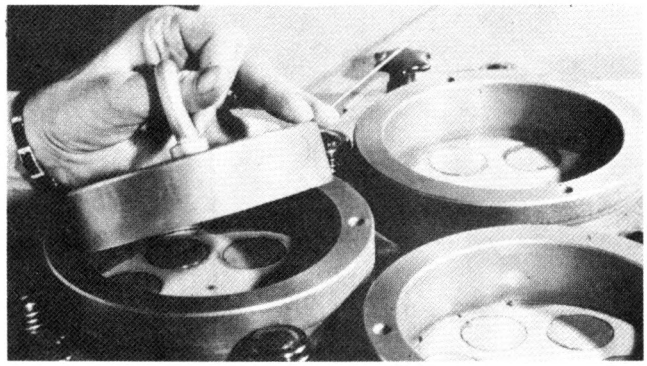
2 - 3 - 1. Un saucisson coûteux

Une rosette de Lyon, ou un salami : voilà ce qu'évoque la matière première de nos circuits intégrés. Ce barreau de silicium de modeste apparence est en réalité un *monocristal* fabriqué par *tirage* : un petit cristal de Si est fixé au bas d'une baguette verticale qui vient toucher un bain de Si en fusion. On la remonte très doucement en tournant, durant plusieurs heures, et la structure cristalline accrochée à celle du germe se prolongera



tout au long du cristal avec un minimum de défauts (mais quelques défauts cependant : dislocations, atomes manquants ou en trop, rares impuretés). Le raffinage du silicium est fait par la méthode de *fusion de zone flottante* que nous avons déjà illustrée.

Une scie diamantée va maintenant découper des *tranches* (ou *plaquettes* ou *rondelles*) de saucisson de Si (anglais *wafers*). Epaisses d'un quart de millimètre seulement, elles seront polies optiquement à l'abrasif fin : notre champ de travail est prêt, il a un diamètre de cinq à sept centimètres environ, limite courante du savoir-faire.

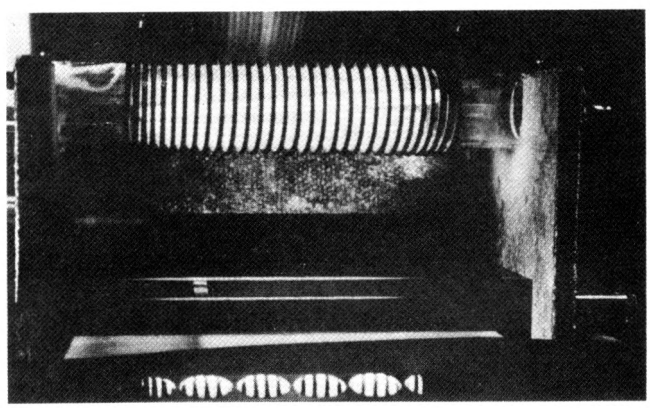


2 - 3 - 2. Epitaxie et diffusions

Nous allons maintenant suivre tout du long ce qui se passe sur une microscopique zone de la tranche (de l'ordre du dixième de millimètre de côté) où nous désirons faire apparaître un transistor et une résistance, par exemple. Mais n'oublions pas que sur une plaquette nous allons fabriquer simultanément plus d'une centaine de pastilles, et que la zone qui nous intéresse n'est qu'un minuscule point sur une quelconque d'entre elles.

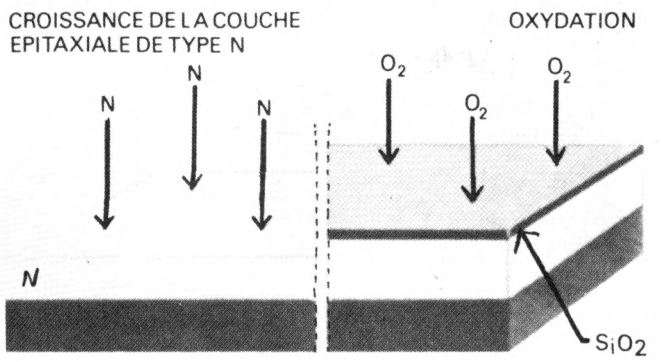
Et c'est à partir d'ici que va entrer en jeu le procédé *planar*, dont nous avons déjà cité le nom, et qui, depuis 1960 où il servit pour la première fois à fabriquer des transistors (plusieurs centaines identiques sur une même plaquette), est devenu — avec d'innombrables variantes — la méthode de base de fabrication des circuits intégrés.

Première étape, *l'épitaxie*. Dans un four tubulaire à haute température, nous plaçons une demi-douzaine de tranches découpées dans un monocristal de Si dopé, de type P par exemple.



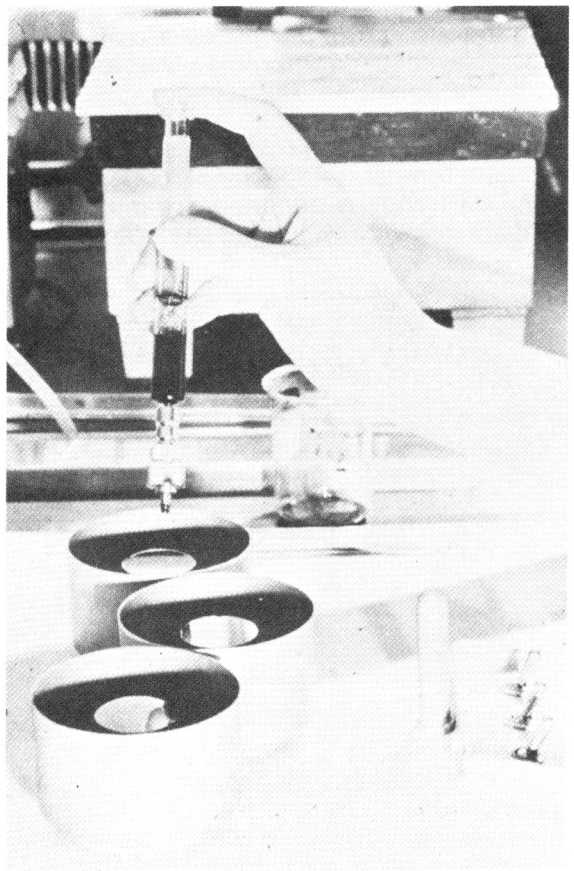
Dans une nacelle, de la poudre de Si dopée N est également introduite : placée dans une zone un peu plus chaude du four, elle se vaporise progressivement et les vapeurs viennent se cristalliser sur la surface polie des plaquettes, formant un

réseau cristallin qui s'accroche sur celui du *substrat* de type P (la plaquette) : c'est l'épitaxie.



Une fois la croissance de la couche terminée, on fait entrer de l'oxygène dans le four, et la surface s'oxyde sur une épaisseur de l'ordre du micron, se recouvrant ainsi d'une pellicule isolante de silice. Nous ne nous refusons rien, notre bijou électronique est revêtu de quartz !

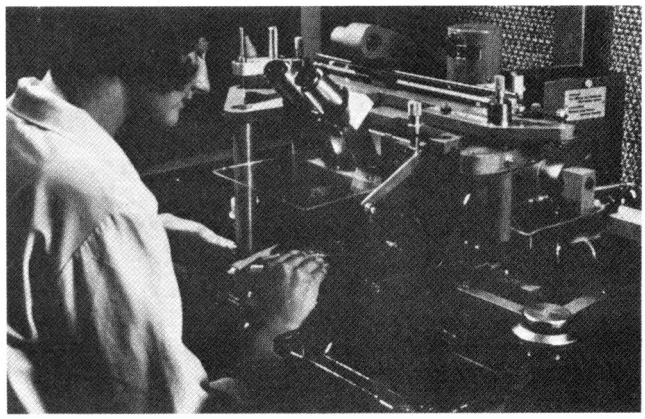
Opération suivante, l'émulsion photographique. La plaquette est montée sur touret tournant à plusieurs milliers de tours/minute. Armée d'une seringue sans aiguille, une opératrice laisse tomber en son centre une goutte de résine sensible,



le *photoresist*, instantanément étalée en fine couche par la force centrifuge : le wafer est prêt, et l'on passe au premier masquage.

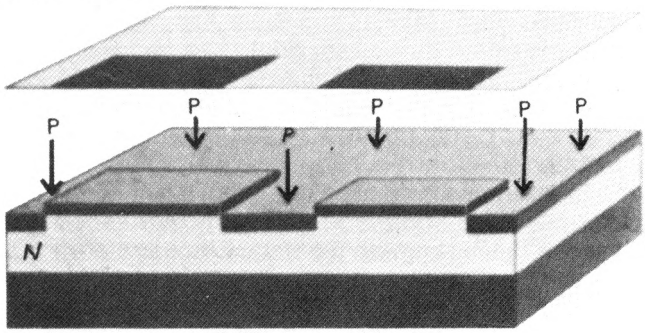
Réalisés par des méthodes dont nous parlerons plus loin, les *masques* (on en utilise en général de quatre à sept selon les types de fabrication) portent sous forme de dessins microscopiques le menu détail du dessin des éléments de nos composants électroniques, tels que base, émetteur, collecteur et connexions d'un transistor, qui seront réalisés tour à tour. Confection et positionnement des masques successifs sur la rondelle sont des opérations de haute précision où tout se mesure en microns, et se vérifie au microscope ou par une caméra de télévision à fort grossissement, avec des points ou des croix de repérage comme dans le travail d'impression en plusieurs couleurs. Un décalage, et l'image est brouillée, les circuits intégrés bons à mettre au panier.

Premier masquage donc, et exposition à une forte source de rayons actiniques : là où le masque est transparent, le photoresist deviendra soluble au cours du développement qui va suivre.



Aspersion d'un solvant, et seul restera sur la tranche le photoresist qui était sous les parties opaques du masque. C'est alors que nous allons faire intervenir un dangereux acide, le seul capable d'attaquer le quartz, *l'acide fluorhydrique*.

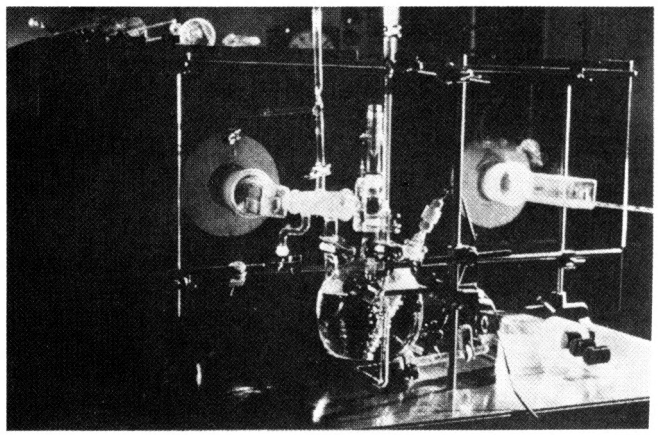
MASQUAGE ET ATTAQUE



PREMIERE DIFFUSION

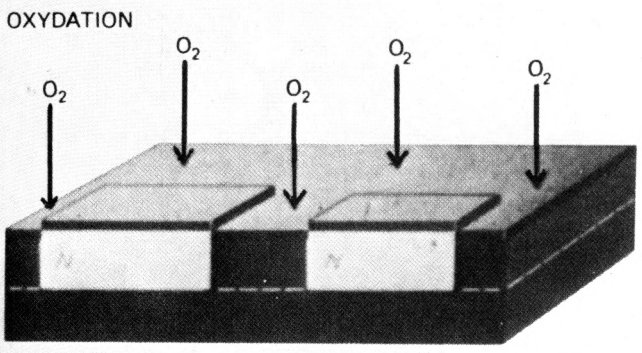
Mais malgré son agressivité, il ne peut rien contre le photoresist : seule la silice à nu se dissout, et l'opération est arrêtée dès que le silicium est à découvert. Enfin, un solvant d'une autre nature enlève le reste du photoresist qui a fini de jouer son rôle, et nous obtenons une rondelle recouverte d'une fine pellicule de silice découpée en forme d'îlots de taille et de forme variées.

Nous allons maintenant réaliser la première *diffusion*. Plusieurs tranches sont introduites dans un four, et des vapeurs dopantes de type P vont lentement y circuler. Les atomes accepteurs vont pénétrer progressivement à travers les zones



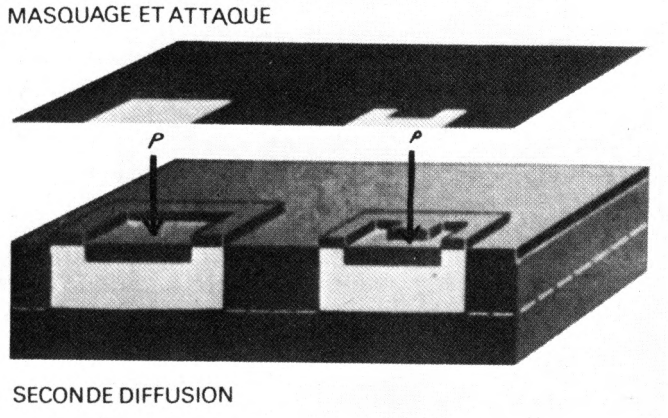
à nu de Si type N, et les transformer de plus en plus profondément en Si type P ; nous arrêterons cette première diffusion au moment où toute la couche épitaxiale sera traversée, sous chaque îlot s'est formé un *caisson d'isolation* qui délimite le contour extérieur d'un composant.

Etape suivante, entrée d'oxygène et à nouveau couche de silice. Deuxième masquage : cette fois par exposition, développement et attaque sélec-

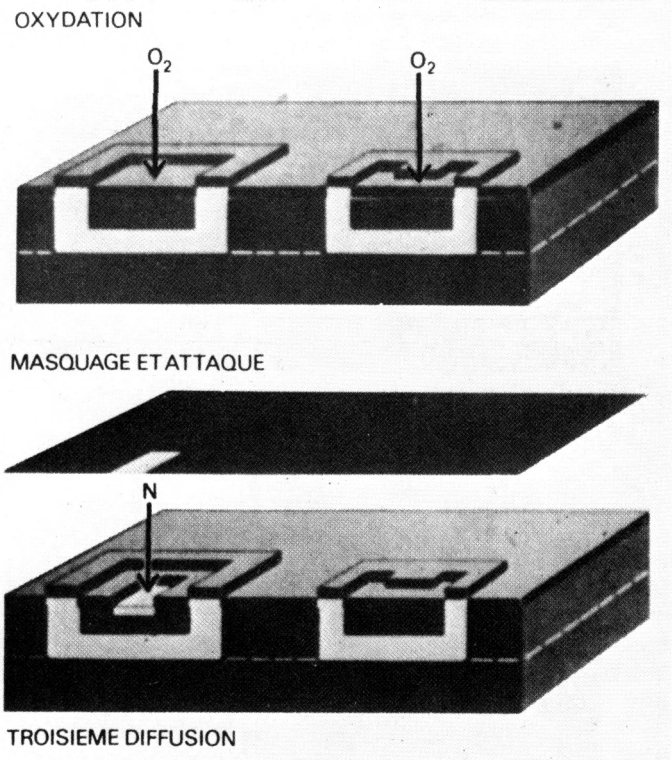


tive comme précédemment, nous allons ouvrir les fenêtres dans la couche de silice.

Deuxième diffusion, toujours avec des vapeurs P, mais cette fois notre diffusion est soigneusement contrôlée pour s'arrêter avant d'avoir traversé la couche épitaxiale. C'est ici que nous voyons apparaître une nouvelle notion : outre la *géométrie horizontale* des composants définie par les dessins des masques, nous contrôlons la *géométrie verticale* définie par la profondeur des diffusions.



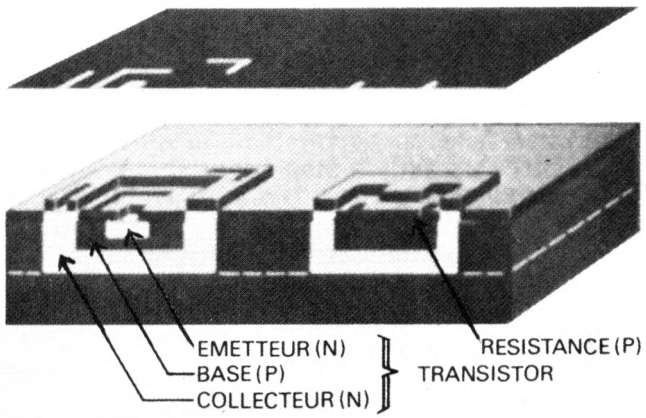
Ainsi faisons-nous «pousser» tridimensionnellement nos composants au sein même du monocristal de semi conducteur, en une succession patiente de petites étapes qui se déroulent dans des salles étanches, hors poussières, où les techniciens des deux sexes évoluent en toque blanche dans de bizarres accoutrements, patauds avec leurs bottes de tissu couvre-chaussures qui leur donnent l'aspect d'oursons blancs... penchés sur des microscopes ou des micro-manipulateurs.

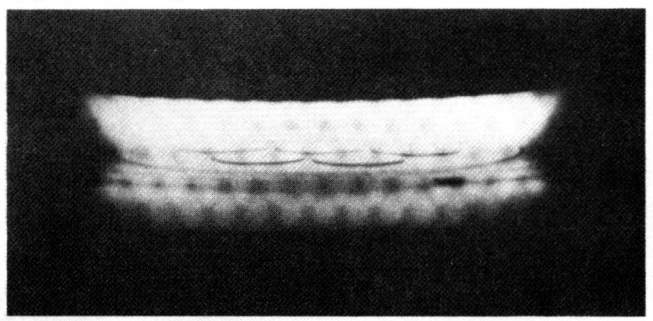


A nouveau oxydation et couche de silice, puis le troisième masque entre en jeu pour la troisième diffusion. Changement maintenant : par les petites fenêtres que nous venons d'ouvrir, diffusion contrôlée à faible profondeur, mais de type N cette fois. C'est la dernière, car maintenant nos composants sont entièrement constitués, tel le transistor situé à gauche de nos croquis, et dont le sandwich s'est développé dans l'épaisseur du cristal. Quant à la résistance qui est à droite, elle était déjà terminée à l'étape précédente, puisque de structure plus simple, et n'a pas subi de modification. Oxydation générale, et nous allons passer à la finition.

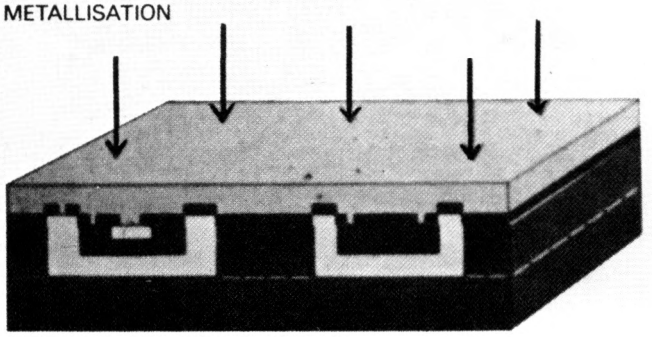
Pour rendre notre circuit intégré opérationnel, il reste à interconnecter les composants entre eux, en prévoyant les connexions externes de la puce. Tout ceci est encore réalisé par masquage. Le

MASQUAGE ET ATTAQUE





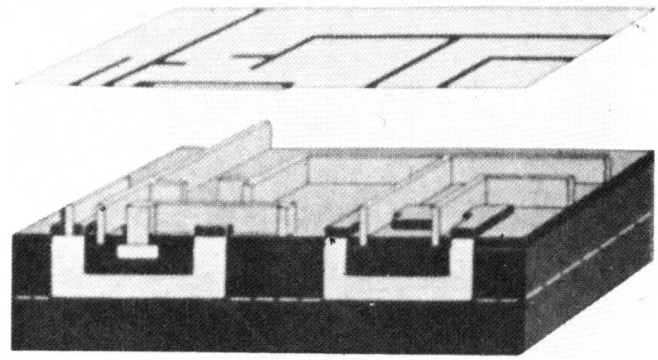
METALLISATION



quatrième masque va nous permettre de percer à travers la silice de petits orifices situés sur les émetteurs, bases, collecteurs, extrêmités des résistances, des diodes, etc. Après attaque et nettoyage, les rondelles sont placées dans une cloche à vide, et une quantité d'aluminium soigneusement pesée est évaporée dans le haut de la cloche, et vient se déposer en fine couche sur les tranches. Cette fois, tout est interconnecté, un peu trop même ! Ce serait le court-circuit total...

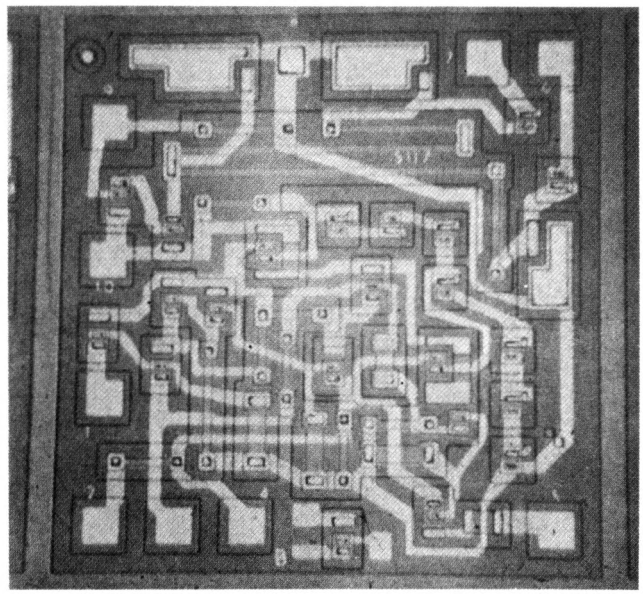
Bien sûr, un cinquième et dernier masquage va nous tirer d'affaire. Eliminant plus des neuf

MASQUAGE ET ATTAQUE

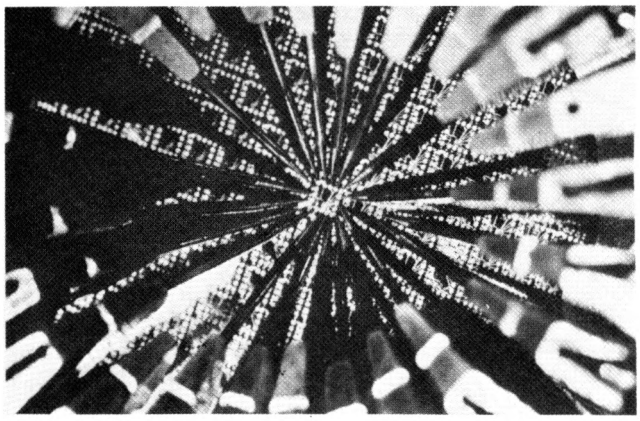


dixièmes de l'aluminium, il ne va en laisser subsister que d'étroits rubans (dont la hauteur est très exagérée pour la lisibilité du croquis) zigzaguant sur les puces pour relier entre eux les composants, et s'élargissant en plots ronds ou carrés sur les bords pour permettre de relier notre circuit au monde extérieur.

Le traitement de notre rondelle est terminé, et nous allons maintenant passer aux *tests*. En effet,

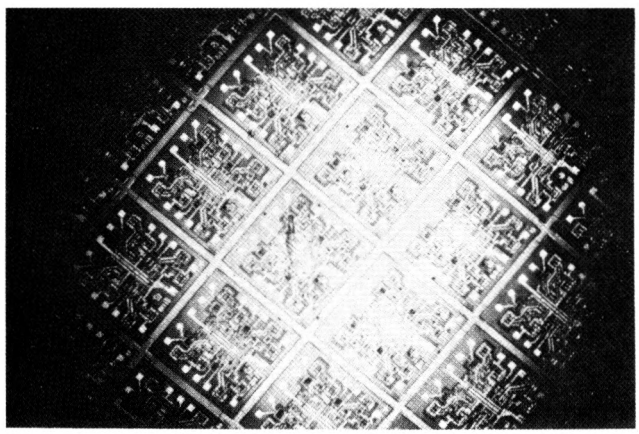


force est d'avouer qu'il y a un déchet de fabrication certain. Le processus est assez délicat, et complexe. Une poussière, un défaut du cristal

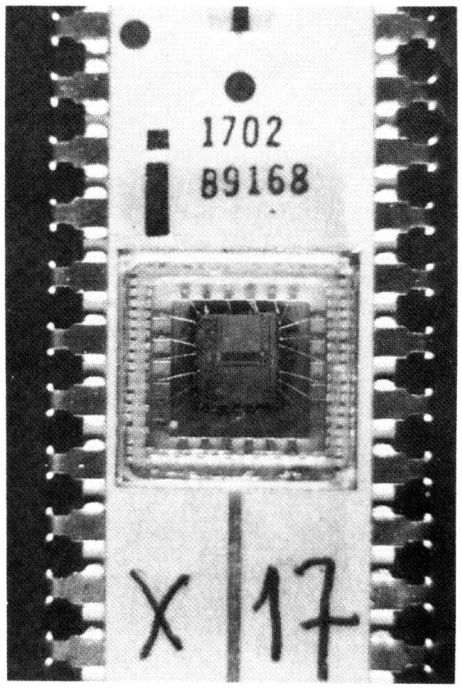


tombant à un mauvais endroit, une opération imparfaite, une contamination, et un ou plusieurs circuits intégrés comporteront des défauts. On présente donc la rondelle à un testeur, quelques dizaines d'aiguilles à ressort viennent faire contact sur les plots périphériques d'une puce, l'appareil mesure rapidement les paramètres du circuit, le marque d'une touche d'encre s'il est hors tolérances, puis passe à la puce suivante.

Une fois les tests terminés, une pointe de diamant permet de détacher les pastilles les unes



des autres, et les bonnes sont brasées chacune dans le fond d'un boîtier (ce qui assure un bon contact thermique pour évacuation de la chaleur pendant le fonctionnement). Ensuite, sous une loupe binoculaire, une opératrice à l'aide d'un minuscule outil ultrasonique soude par *thermocompression* les fils d'or plus fins que des cheveux qui relieront les plots de la puce aux broches extérieures du boîtier. Encapsulage de protection, et le circuit intégré est terminé.



2 - 3 - 3. SOS

Le lecteur curieux aura sans doute accroché au passage sur un détail gênant : partout nous avons considéré les composants du circuit intégré comme indépendants les uns des autres, hormis les liaisons dues aux connexions d'aluminium. Or, cependant, tous sont à l'intérieur d'un substrat semiconducteur, et ne sont par conséquent pas isolés les uns des autres. Comment notre circuit intégré peut-il donc fonctionner ?

Revenons en arrière, à notre premier masquage, celui qui a réalisé les caissons d'isolation : la clé est là. Chaque caisson forme avec le substrat une jonction. Polarisons de façon inverse avec une faible tension l'ensemble de ces jonctions, et le tour est joué, chaque caisson est devenu *isolant*, comme toute bonne jonction inversement polarisée, et justifie pleinement son nom... Avouez que l'astuce est géniale ! Bien que monolithique, le circuit intégré se comporte donc comme des centaines de composants séparés.

Quant on veut pousser au maximum les performances, cette isolation n'est pas suffisante, et l'on cherche à séparer physiquement les composants. De nombreuses méthodes ont été utilisées dans ce but. Jadis, on a utilisé la méthode des *connexions poutrelles* (beam lead). Après une forte métallisation, d'or par exemple, on retournait la rondelle et par masquage et attaque sur la face

arrière on détachait les composants les uns des autres : c'est le treillis des connexions qui tenait tout en place ! Actuellement, la technologie de pointe *SOS (silicon on sapphire, silicium sur saphir)* atteint le même résultat d'une façon un peu moins inquiétante. Créée pour l'aéro-spatial, elle commence à être utilisée en informatique, en particulier pour certains mini-ordinateurs. Cette fois, le substrat de départ est un monocristal parfaitement isolant, dont le réseau correspond à celui du silicium : on utilise le *corindon*, Al_2O_3, qui est de l'oxyde d'aluminium (*alumine*) cristallisé. Minéral qui ne le cède en dureté qu'au diamant (il est très employé comme abrasif), le corindon lorsqu'il est teinté par des impuretés métalliques dans la nature s'appelle... rubis ou saphir. Encore la joaillerie électronique... Sur notre rondelle de corindon, par épitaxie nous déposerons du silicium, et après tous les masquages successifs nous pourrons obtenir des composants séparés les uns des autres par un substrat parfaitement isolant, bien que l'ensemble soit également parfaitement monolithique !

2 - 3 - 4. Masques et artistes

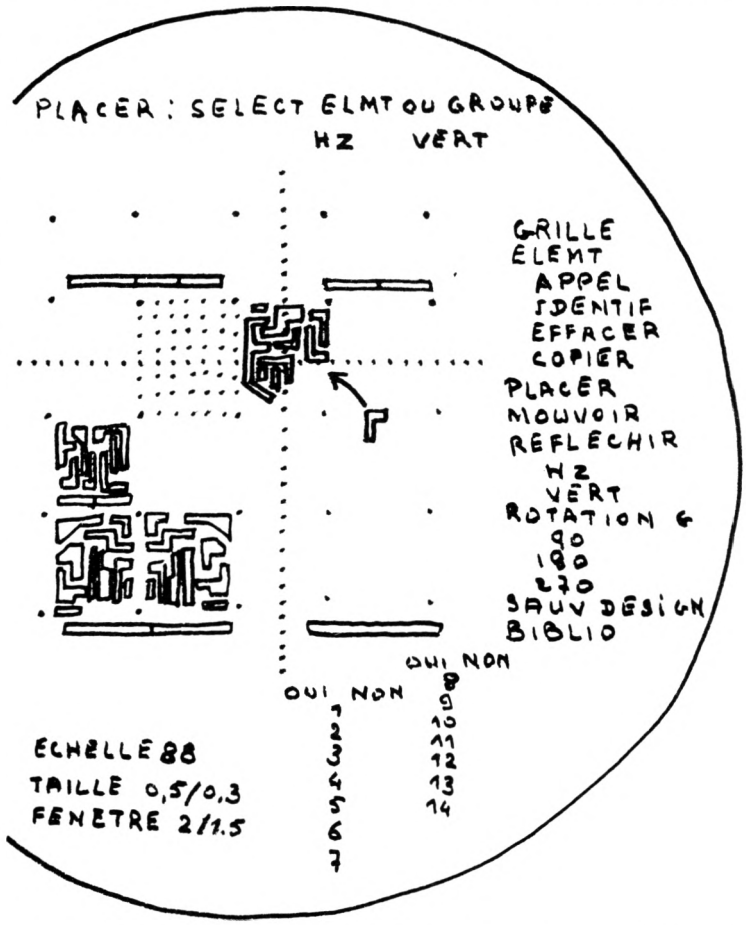
Tout le travail de conception d'un circuit intégré monolithique se concrétise finalement sous la forme du jeu de masques nécessaire à sa fabrication. Le schéma théorique, à travers un *art work* patient, va prendre forme sous les doigts des dessinateurs et aboutira à la découpe de feuilles plastiques spéciales en *rubylith*, destinées à l'exécution des clichés photographiques que sont les masques.

En fait, plusieurs méthodes de conception coexistent actuellement. Une fois qu'ont été définies, mises au point, les spécifications logiques et électriques du circuit intégré, on réalise *l'implantation* de la pastille sur un dessin de dimensions 500 à 1000 fois plus grandes que l'original (si l'on peut dire, car ce dernier — la puce — n'existe pas encore !). Superposés en couleurs différentes sur ce chef d'œuvre labyrinthique, nous trouvons la totalité des masques nécessaires, ce qui permet de les repérer parfaitement les uns par rapport aux autres. En pratique, l'opération pourra se faire sur papier, plastique... ou écran cathodique relié à un ordinateur.

2 - 3 - 5. CAO

C'est ici que peut intervenir la *CAO, conception assistée par ordinateur (CAD, computer aided design)*, par le moyen de tables traçantes, ou de visualisations, et bien entendu des centaines de sous-programmes adéquats organisés en un système de CAO spécialisé.

On pourra donc en particulier composer les masques de circuits intégrés sur un écran cathodique, par assemblage d'éléments de dessin stockés sur mémoire à accès sélectif, et appelés au clavier par leur indicatif. Ils apparaissent sur l'écran à l'endroit indiqué par un attouchement du crayon photoélectrique sur une liste des numéros d'emplacements (voir croquis). Les éléments non standard sont dessinés à la main sur l'écran, puis,



remorqués par le crayon, viennent se loger en bonne place. A droite de l'écran figure une liste de commandes permettant d'indiquer au programme qui est dans l'unité centrale les manœuvres qu'on veut effectuer. Posons l'extrémité de la «plume de lumière» sur SAUVDESIGN : le programme interrompu repère que nous visons l'avant-dernier ordre de la liste, et que nous désirons par conséquent sauver notre création toute fraîche en l'inscrivant sur la mémoire à disques.

Notons au passage que si l'écran a une résolution de 1.000 lignes à 1.000 points par ligne, nous serions amenés à stocker un million de bits pour une image (un bit vaudra 1 si l'écran est lumineux en ce point, 0 s'il est éteint) : c'est peu dans la mémoire à disques, mais beaucoup pour la mémoire centrale du calculateur où l'image doit bien entendu être stockée pour guider le pinceau d'électrons du tube cathodique. On s'en tirera donc autrement : chaque image correspondant à

un masque est composée de quelques milliers de segments de droites, ou «vecteurs». Pour définir l'un d'eux, il suffira de stocker la position de ses deux extrémités, soit deux nombres de 10 bits chacun (2_{10} = 1.024, ce qui couvre la résolution de tube) (18). S'il y a 18.000 vecteurs, cela fera 160.000 bits ou 20 K octets en tout, plus les coordonnées des points de la grille de fond optionnelle, des axes, les codes des lettres du texte, etc., ce qui devient on ne peut plus raisonnable.

Une fois notre dessin sauvé, un autre programme le relira ultérieurement sur les disques et, en l'analysant vecteur par vecteur, calculera une longue série d'ordres et les perforera à la queue leu leu sur un hectomètre de bande de papier. Il ne restera qu'à alimenter à partir de cette bande perforée une machine à commande numérique : elle découpera un des six masques servant à fabriquer notre circuit intégré, équivalent peut-être à quatre ou cinq cents transistors, diodes et résistances.

De quatre mois, le délai de conception est ramené à trois semaines, et seul l'ordinateur a pu le permettre.

2 - 3 - 6. La course de la puce et de l'ordinateur

Cependant, une véritable course s'est produite entre les possibilités de la machine et les exigences toujours accrues des ingénieurs qui, maîtrisant de mieux en mieux la technologie de fabrication, sont en mesure d'augmenter à la fois la densité en composants au millimètre carré et les surfaces des puces. En effet, d'une part techniques photographiques et de masquage se perfectionnent, et d'autre part le nombre de défauts, taches, poussières par unité de surface diminue, permettant d'agrandir les pastilles. C'est ici qu'intervient un paramètre essentiel en fabrication de circuits intégrés, le *rendement (yield)*, défini comme le pourcentage de bonnes puces sur la rondelle par rapport au nombre de puces total. Supposons que dessinant aujourd'hui un circuit intégré, après mise en fabrication le rendement se révèle de 90 % : bravo direz-vous ? Eh bien, non. Cela veut dire que nous avons été trop timides, et que vu notre savoir faire technologique nous aurions pu concevoir un circuit beaucoup plus complexe, regroupant les fonctions de plusieurs autres, avec un prix au transistor unitaire bien plus faible. Tout l'art consiste donc à savoir à un moment de combien surpasser son propre talent. Le rendement sur les premières rondelles sera peut-être piteux, de quelques pour cent, mais la fabrication se rodant progressivement nous atteindrons les 30 % signe d'un bon compromis...

(18) Dans le cas d'un contour, on économisera la moitié des nombres.

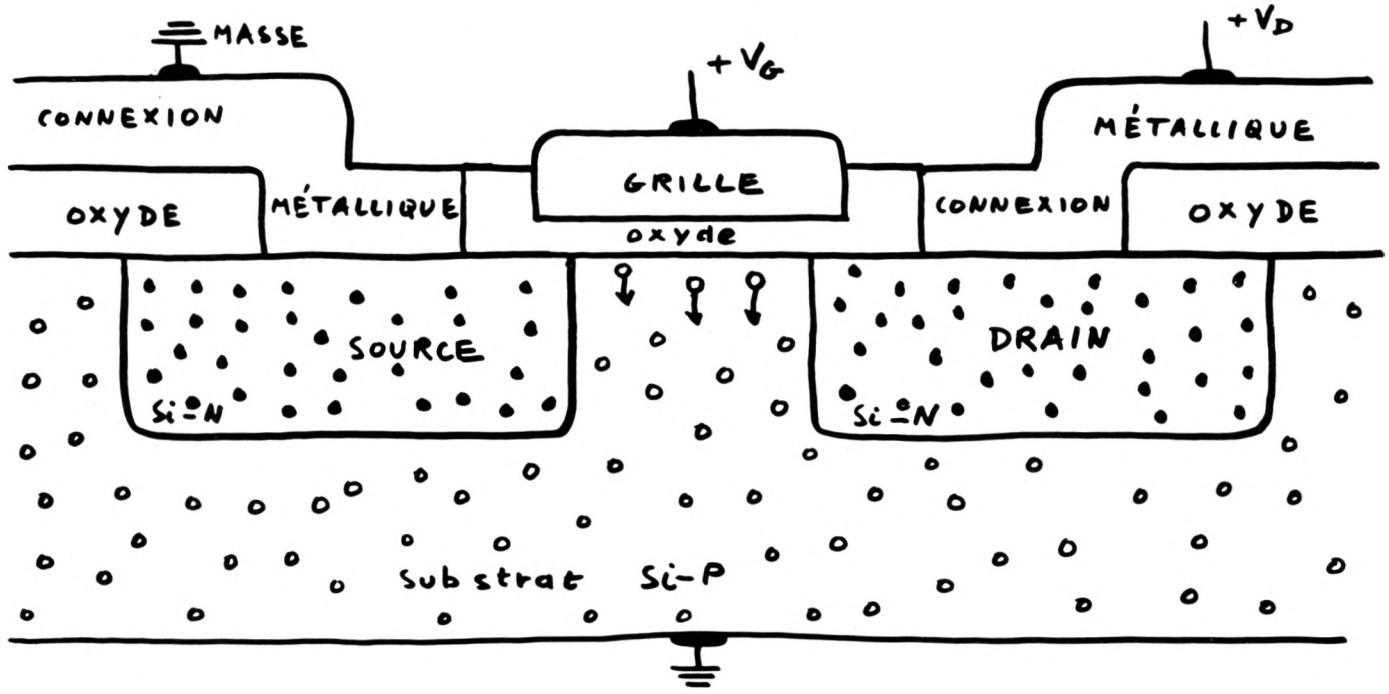
Mais revenons à la course de la puce et de l'ordinateur : de 1971 à 1974, la surface des puces est passée de 10 à 30 mm², le nombre total de coordonnées de 50.000 à 150.000, et maintenant les dessinateurs marchent à quatre pattes sur des dessins de cinq mètres sur trois. Patiemment, ils recherchent les erreurs de tous genres (1 à 10 points sur mille avec les équipements et les techniques les plus récents) qui parsèment le labyrinthe, semaine après semaine s'il le faut. Ailleurs, des tables traçantes géantes de plusieurs mètres, solitaires dans une ambiance de véhicule spatial, brodent inlassablement leur dentelle multicolore avec des sifflements légers, avançant, reculant, zigzaguant, festonnant avec une rapidité et une précision surhumaine, issues de l'ordinateur, et qui rendent pensif le visiteur le plus blasé. Mais bien sûr, les éléments initiaux du dessin ont d'abord été *digitalisés*, des opératrices suivant patiemment point par point le dessin original à l'aide d'un curseur relié à une machine qui enregistre les coordonnées numériques.

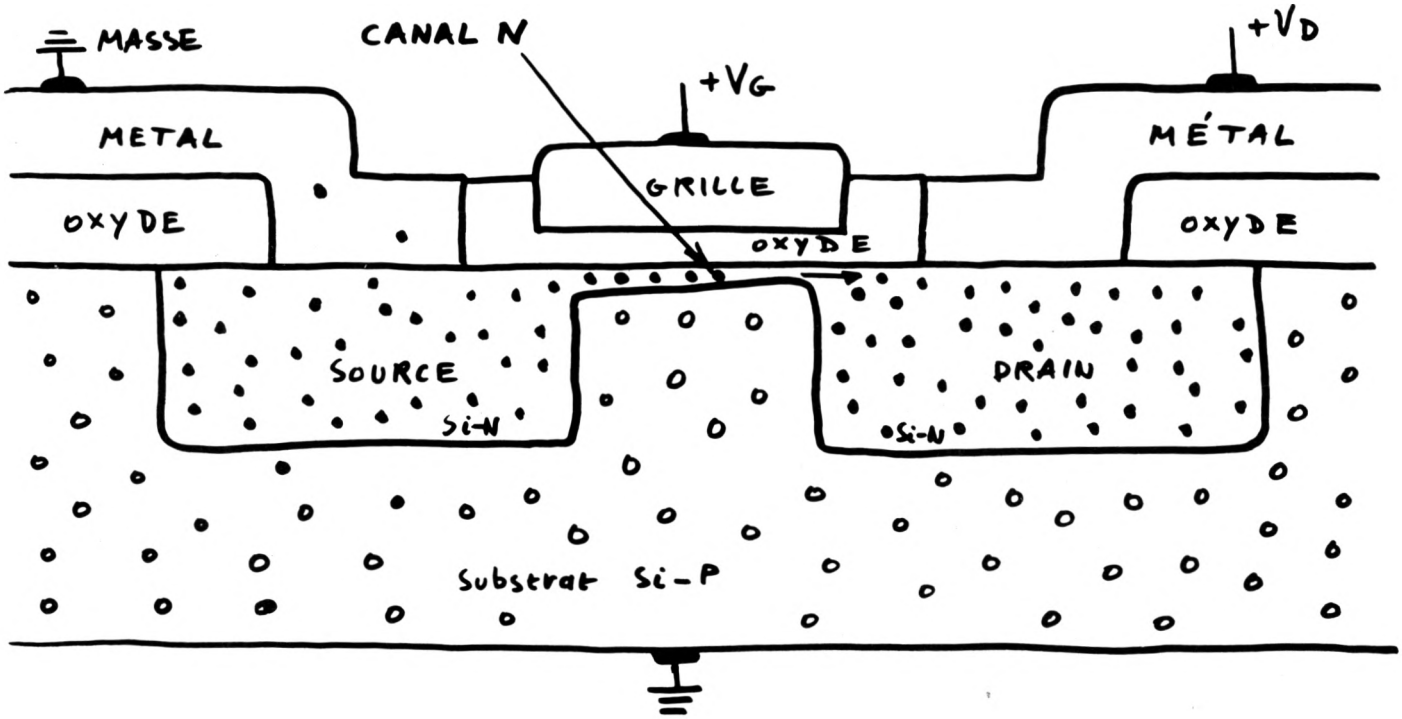
Au fil des progrès technologiques d'un côté ou de l'autre de la barrière — celui de la puce ou celui de l'ordinateur — on utilise donc plus ou moins directement ce dernier, mais de toute façon d'ores et déjà on peut dire que l'ordinateur, ici encore, contribue à faire naître d'autres ordinateurs plus modernes qu'il ne l'est lui-même : nous n'attendons plus qu'un Darwin de la machine pour y voir plus clair pour l'avenir...

2 - 4. ACTE TROIS : LE TRANSISTOR A EFFET DE CHAMP

Le transistor classique, à deux jonctions, ou transistor *bipolaire*, est le plus répandu (en tant que composant encapsulé isolément), et constitue l'élément de base fndamental des circuits intégrés traditionnels, ou *circuits intégrés bipolaires* (comme ceux de la logique TTL) dont nous avons décrit la fabrication en détail. Notons au passage que ce qualificatif de bipolaire (*bipolar* en anglais) est dû au fait que ces transistors utilisent pour transporter le courant les deux types de porteurs simultanément : trous et électrons, positifs et négatifs. On les appelle aussi quelquefois à *injection*, à cause du rôle essentiel que joue l'injection de porteurs dans la base par émetteur.

Mais il existe un autre type de transistors, de plus en plus important chaque année, car c'est lui qui est le composant essentiel des *circuits intégrés MOS*, lesquels servent à réaliser les microprocesseurs... et les mémoires à circuits intégrés. Il s'agit du *transistor à effet de champ*, ou transistor *unipolaire*, et vous l'aurez deviné, il n'utilise qu'un seul type de porteur (angl. : *field effect transistor, FET*). Si nous lui avons réservé la deuxième place dans notre exposé, c'est à la fois pour des raisons chronologiques, et parce qu'il est plus simple que le transistor bipolaire : en particulier, il désarçonnera moins ceux parmi nos lecteurs





Transistor à effet de champ (technologie MOS canal-n classique)

électrons trous

Source et drain sont diffusés dans les premières étapes de fabrication, comme en bipolaire.

qui se souviennent de la bonne vieille triode. Le principe en est en fait assez ancien : c'est vers 1925 que Julius Edgar Lilienfeld découvrit qu'un champ électrique appliqué perpendiculairement à un cristal de silicium pouvait affecter sa conductivité. Bien que deux brevets lui aient été attribués en 1926 et 1935, la compréhension à l'époque de la physique des solides et du comportement des films minces ne permit d'arriver à aucune réalisation pratique. Ce n'est que bien plus tard, à la suite des travaux théoriques de William B. Shockley et Walter Schottky en particulier, que le premier FET fut réalisé en 1953, puis enfin en 1960 sous une forme voisine de la configuration actuelle que nous allons décrire.

Si, comme le transistor bipolaire, le transistor à effet de champ comporte trois électrodes, l'une d'entre elles — la *grille*, en général en aluminium, et qui est l'électrode de commande — est isolée des deux autres. Mais alors, direz-vous, comment peut-elle agir ? Par influence à distance, à l'aide de son champ électrique, et c'est là tout le secret du FET.

Considérons le croquis qui figure deux états de notre transistor : bloqué, puis conducteur. On y voit tout d'abord que deux des électrodes la *source* et le *drain*, sont symétriques et constituées par des zones de type N diffusées dans un substrat de type P. Ce qui différenciera la source du drain, ce sont les tensions qui leur sont appliquées : ici, la source sera à la masse (potentiel zéro par définition), le drain étant à un potentiel positif, prêt à attirer vers lui des électrons. Mais en l'absence de tension appliquée à la grille, aucun courant ne peut passer de la source au drain.

Appliquons maintenant une tension de polarisation progressivement croissante à la grille qui va donc prendre une charge positive. Bien qu'elle soit isolée par une très mince couche de silice (le *diélectrique de grille*, d'une épaisseur de l'ordre d'un dixième de micron) le champ électrique positif qui l'entoure va d'abord chasser les trous du substrat à une certaine distance, puisqu'ils sont positifs eux aussi. Mais augmentons encore la tension : à prtir d'une certaine valeur dite tension de *seuil*, la grille attirant de plus en plus fortement les électrons situés à droite et à gauche, il se produit en surface du substrat, juste sous le diélectrique de grille, une *couche d'inversion* de type N (inversion de porteurs) qui va constituer un chemin de passage pour les électrons entre la source et le drain.

Ce chemin est baptisé *canal*, et nous sommes donc en présence d'un transistor MOS *canal n*, dernier cri de la technologie. MOS, signifiant Metal Oxide Silicon (en français métal oxyde silicium) nous rappelle la structure de notre FET : grille en métal, diélectrique en silice et reste du transistor en silicium dopé. Nous verrons que ce qualificatif MOS reste conservé pour des versions de transistors à effet de champ où la grille n'est plus métallique.

Une fois la tension de seuil dépassée, le courant passant à travers le transistor sera proportionnel à la tension de polarisation de la grille, ce qui est une propriété intéressante pour l'amplification.

Supposons maintenant que source et drain soient de type P, et le substrat de type N. En appliquant une polarisation — négative cette fois — à la grille, nous obtiendrons la conduction par création d'un *canal p.* Ce type de transistor MOS est en fait à l'heure actuelle de loin le plus répandu, car le plus facile à fabriquer (nous en reparlerons plus loin). Mais les performances des transistors canal n sont bien meilleures : rappelons-nous que la mobilité des électrons est de deux à trois fois celle des trous, et nous ne nous étonnerons pas que la deuxième génération de microprocesseurs — ceux que nous allons utiliser pour Nanostar — soit réalisée en canal n.

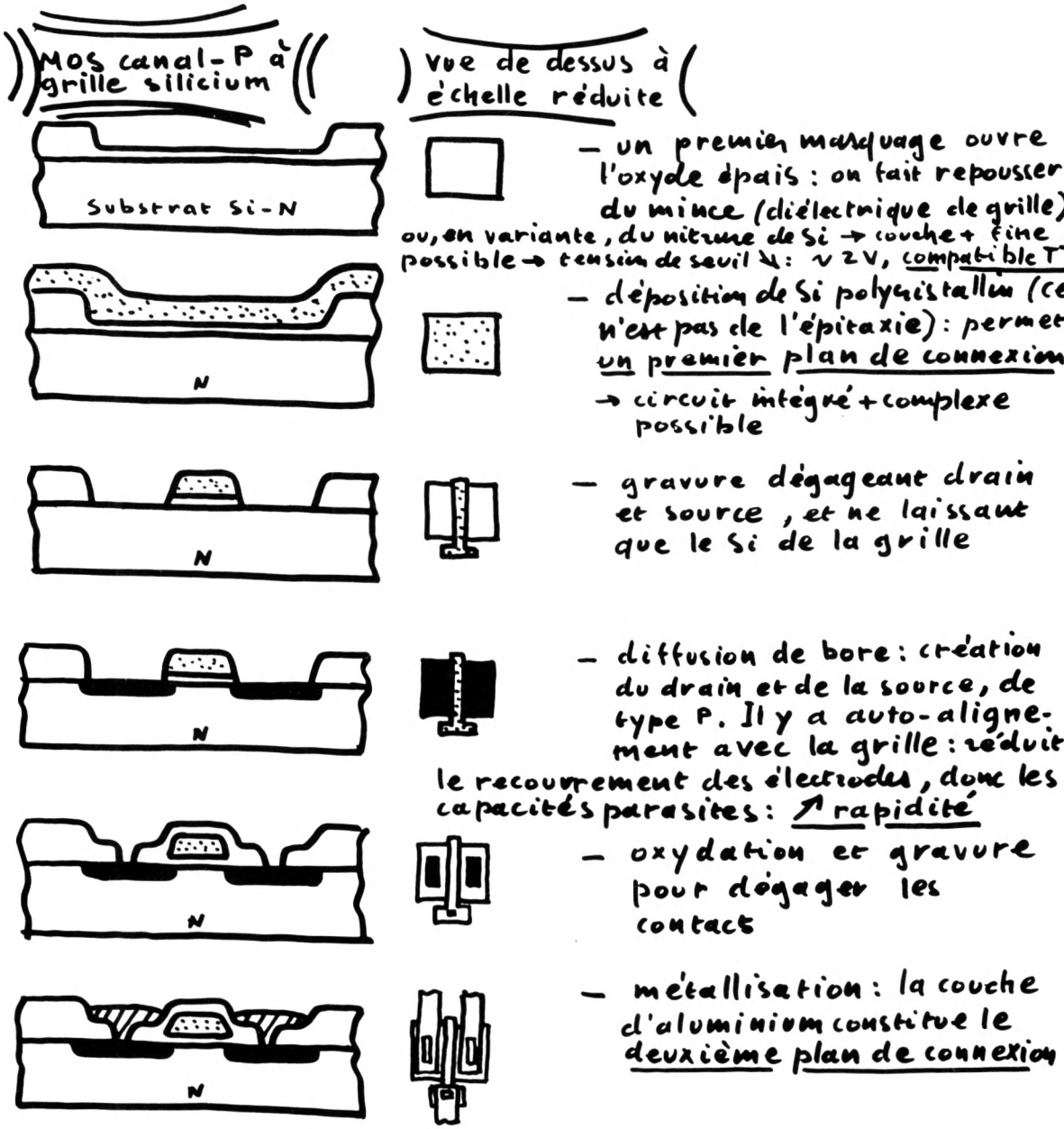
2 - 5. DERNIER ACTE : LES CIRCUITS INTEGRES MOS

Nous avons vu les principes de base de la fabrication des circuits intégrés bipolaires : photogravures et diffusions répétées. Ce sont les mêmes qui seront utilisés ici, mais avec encore plus de fruit, car les transistors MOS se prêtent mieux que les bipolaires au processus *d'intégration* (c'est-à-dire au rassemblement sur une seule pastille de centaines — ou de milliers — de transistors séparés). En effet, nous connaissons maintenant le rôle des caissons d'isolation qui permettent l'intégration en bipolaire. Or, ici, nous n'en aurons plus besoin. Source, drain et canal étant constamment maintenus à une polarisation inverse par rapport au substrat, une jonction isolante entoure complètement notre FET. Il s'ensuit un avantage considérable : l'encombrement périphérique du caisson disparaît, et chaque transistor sera beaucoup plus petit, d'autant plus que sa géométrie verticale est plus simple, ce qui se traduit encore par des dimensions horizontales réduites puisque les couches diffusées successives s'enveloppent les unes les autres comme les pelures d'un oignon. Le canal a en général une largeur inférieure à cinq ou dix microns (nous devrions dire micromètres, mais les habitudes sont difficiles à changer), et les dimensions hors tout du transistor sont de l'ordre d'une cinquantaine de microns.

Côté fabrication l'épitaxie est supprimée, il y a beaucoup moins d'opérations à haute température (les plus critiques) et le nombre d'opérations total (130 environ pour le bipolaire) tombe à trois fois moins. Cependant, le MOS est une technique relativement nouvelle par rapport au bipolaire, très sensible aux contaminations et qu'il faut savoir maîtriser. Revenons par exemple au canal n ; nous avons vu son gain en rapidité, mais ce n'est pas le seul : à caractéristiques égales, la surface d'un transistor MOS canal n est la moitié de celle d'un MOS canal p, d'où des possibilités d'intégration très accrues. Or, il a fallu attendre le début des années 70 pour commencer à maîtriser le canal n. Raison ? La sensibilité du pro-

cédé aux impuretés, en particulier les ions sodium. Or, tous les chimistes vous le diront, «le sodium, il y en a partout»... car cet élément est à la fois très courant et très réactif. Si en cours de fabrication des ions positifs viennent contaminer la surfae du silicium, ils constitueront sous le diélectrique de grille une charge positive tendant à débloquer (intempestivement) le transistor s'il est à canal n. Si c'est un canal p, la grille négative les fera migrer vers l'interface aluminium-silice où ils sont moins gênants, et de toute façon ils ne feront que rendre un peu plus difficile le débloquage du transistor. Voilà comment la nature travaille contre le canal n...

Un très grand nombre de variantes de fabrication existe en MOS (en bipolaire aussi, bien sûr !) :

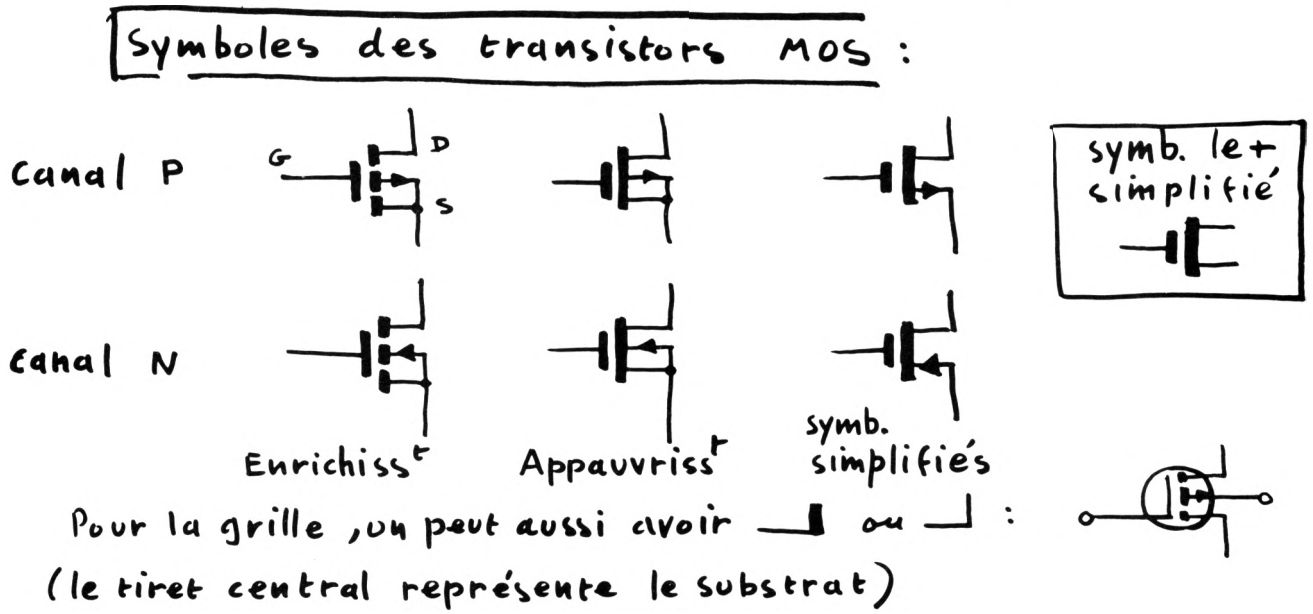


..de là Le MOS canal-N à grille silicium et nitrure: microprocesseurs de seconde génération, en inversant zones N et P ... et en maîtrisant mieux encore le processus de fabrication (contaminations)

— orientation de la taille du cristal de silicium par rapport au réseau (111 ou 100, ceci pour les cristallographes, la coupe 100 réduisant la tension de seuil),
— canal p ou n,
— à *enrichissement* ou *appauvrissement* : en effet, dans les transistors décrits, le canal n'existe pas si la grille n'est pas polarisée. C'est en y appliquant la tension qu'on provoque un enrichissement (en porteurs) qui fait apparaître le canal. Par un processus de diffusion légèrement différent, on peut au contraire obtenir à la fabrication un canal, qui disparaîtra quand on polarisera la grille : ce deuxième type de transistor est dit à appauvrissement (*depletion*), et est donc débloqué au repos,
— électrode de grille en aluminium, silicium polycristallin, molybdène, tungstène, etc. La grille en silicium est un procédé important et astucieux que nous détaillons dans des croquis, car c'est en particulier lui qui — conjointement avec le canal n — a permis les microprocesseurs de deuxième génération, fonctionnant à deux mégahertz.

exemple, la bascule maître-esclave (le JK flip-flop) se ramène aisément à un assemblage de NAND. Et à vrai dire un seul type de boîtier pourrait suffire, comme le quadruple NAND à 14 broches : cependant, pour la convenance et l'économie, les catalogues de composants électroniques nous offrent des combinaisons plus variées. On trouve en *DIP* 14 broches (abréviation familière de *dual in line package*, conditionnement à deux rangées de broches) les circuits intégrés suivants :

— 4 portes NAND à 2 entrées
— 4 portes NOR à 2 entrées
— 2 JK
— 2 bascules du type K
— 6 inverseurs
— 3 portes NAND à 3 entrées
— 2 portes NAND à 4 entrées
— 1 porte NAND à 8 entrées
— même chose pour NOR ;
et depuis peu de temps :
— 4 portes AND ou OR à 2 entrées ;



2 - 6. LES BOITIERS A UN FRANC

2 - 6 - 1. Le quadruple NAND

A travers tous ces efforts, la logique abstraite et pure des portes ET, NAND, OR et autres, a fini par se concrétiser sous la forme de ces boîtiers noirs rectangulaires, à deux rangées de pattes, qui sont l'emballage plastique de nos circuits intégrés, les briques logiques à l'aide desquelles nous allons bâtir nos systèmes. En reliant entre elles par les conducteurs de nos circuits imprimés les broches des boîtiers, nous pourrons résoudre tout problème de logique combinatoire *et* séquentielle. Par tout ceci pour un prix n'excédant guère un franc. Si nous allons plus loin, les fonctions deviennent très puissantes :

— un décodeur entier BCD (exemple vu)
— un registre mémoire de 8 bits sur lequel il est possible de faire des opérations de :

- chargement
- lecture
- décalage à droite ou à gauche

— un additionneur complet sur 4 bits
— un comparateur de 2 mots de 4 bits
— un générateur de parité sur un mot de 8 bits.

Par contre, ces boîtiers ont généralement de 18 à 24 pattes.

2 - 6 - 2. Les familles

On s'en rend compte, les combinaisons possibles (même élémentaires) des portes de base sont très nombreuses, de nouvelles apparaissent sans cesse sur le marché. Une standardisation s'est donc faite sur le plan des tensions électriques qui représentent les valeurs logiques, et des tensions d'alimentation. Ainsi, on a vu se créer des *familles*, et au sein d'une quelconque d'entre elles les composants sont entièrement compatibles, de sorte que pour les assembler seul l'aspect logique du problème compte : on est débarrassé de toute considération électronique. Après avoir manié les tables de vérité, il ne reste plus qu'à prendre en main le fer à souder (ou tout au moins c'est presque vrai...).

Le lien qui unit chacune de ces familles est la technologie employée, puisque c'est elle qui va conditionner les valeurs des tensions utilisées. Au cours du temps, les familles se succèdent donc avec un recouvrement plus ou moins grand. Essayons d'en donner un aperçu (plus ou moins) chronologique :

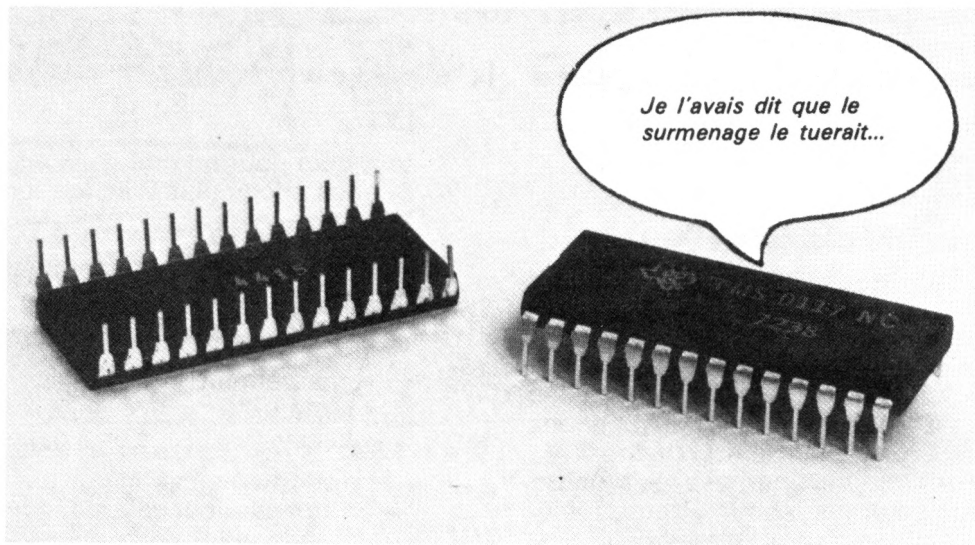
— **RTL** : Rien à voir avec la station de radiodiffusion ! Le sigle signifie *resistor transistor logic (logique à résistances et transistors).* C'est la plus ancienne, et à l'époque (années 50) les composants utilisés étaient *discrets*, c'est-à-dire sous conditionnement individuel (le petit chapeau haut de forme métallique des transistors bipolaires avec leurs trois pattes, les minuscules cylindres des résistances avec les anneaux codés colorés indiquant leur valeur, et l'anneau de tolérances : or + 5 %, argent + 10 %, rien + 20 %).

— **DTL** : *Diode transistor logic (logique à diodes et transistors)* qui a eu son heure de gloire.

— **TTL** : *Transistor transistor logic (logique à transistors... et toujours transistors).* C'est la plus répandue actuellement : sa technologie utilise les circuits intégrés monolithiques bipolaires, et les listes de boîtiers que nous avons citées sont disponibles en TTL.

— **ECL** : *Emitter coupled logic (logique à émetteur couplé).* Ici, un nouveau concept apparaît. Les familles précédentes (et les suivantes) utilisent la *logique saturée,* c'est-à-dire qu'un transistor (ou une diode) sera soit bloqué soit pleinement conducteur : fonctionnement par tout ou rien, avec basculement franc. Or, nous l'avons vu, en passant d'un état à l'autre des populations entières de porteurs doivent changer, et les intensités de courant possibles en pratique étant limitées, le basculement total demandera un certain temps, le *temps de commutation (switching time).* Plus il est court, plus la logique est rapide (voir «*Un fil d'Ariane*», volume 1, les *délais de propagation* et le temps d'addition dans un additionneur binaire du type half-adder en fonction des *couches de logique*).

En ECL, à l'aide d'une liaison particulière de l'émetteur dans les schémas électroniques, les transistors fonctionnent en *logique non saturée,* en deux points plus proches l'un de l'autre que les deux extrémités du basculement, et les temps de commutation (au moment où l'ECL est apparu) sont de l'ordre de dix fois plus courts. Actuellement le temps de propagation d'un signal est de l'ordre de 10 ns pour le TTL standard, et de 2,5 ns pour l'ECL. Cependant, cette technologie est difficile à mettre en œuvre : en particulier les longueurs de câblage



entrent en compte pour la mise au point de la logique, ce qui n'est heureusement pas le cas en saturé. Et certaines technologies récentes, telle l'avant dernière de notre liste, battent sérieusement en brèche l'ECL pour de nombreuses applications.

— **MOS canal p et canal n** : Ce sont les nouveaux venus prometteurs dont nous avons parlé : ils permettent donc une intégration beaucoup plus poussée que les autres technologies (c.à.d. davantage de transistors par puce), mais leur rapidité au début était nettement plus faible que celle du bipolaire. Heureusement avec le canal n, puis la grille silicium, puis d'autres améliorations à venir, la vitesse s'accroît régulièrement sans limitation proche en vue.

— **C/MOS** *(complementary MOS, MOS complémentaire)* : Dans cette technologie, on diffuse sur une même puce des FET canal p et canal n. Il est ainsi possible de réaliser des circuits logiques dotés d'une consommation extrêmement faible. Le C/MOS est pour l'instant surtout employé dans les domaines où cet avantage est primordial : les montres bracelet électroniques par exemple.

— **Schottky TTL** : Nouvelle version rapide du TTL, rajeunissement du bipolaire, lequel loin d'être encore au bout de son évolution se défend vigoureusement contre le MOS ! le Schottky avec un temps de propagation de 3 à 5 ns va permettre de réaliser des microprocesseurs aussi rapides que les minis actuels : il y a déjà sur le marché des *tranches fonctionnelles* de processeurs larges de deux bits, à grouper en nombre de n/2, 2 étant le nombre de bits du mot de la machine qu'on désire utiliser (mais il faut créer de toutes pièces le jeu d'instructions par microprogrammation).

— **I²L** : *(integrated injection logic). La logique à injection I²L,* en expérimentation depuis trois ans, puis en production en faible quantité, a fait tout récemment de brusques progrès : une porte I²L occupe maintenant une surface quatre à dix fois inférieure à une porte TTL. Quant à la consommation elle est toujours aussi microscopique, bien inférieure à un microwatt par élément. Voilà donc un concurrent sérieux du C/MOS pour les montres, et du MOS canal n : sa vitesse est intermédiaire entre celle de ce dernier et celle du Schottky.

Soyons sûr que dans l'avenir cette liste ne va pas cesser de s'allonger !

2 - 6 - 3. Les familles ennemies

Et entre familles, que se passe-t-il ? Avons-nous à nous en préoccuper ?

La réponse est oui, car il arrive qu'on ait besoin de faire parler des membres de l'une à des membres d'une autre : par exemple un mocroprocesseur MOS à des DIP d'interfaces TTL...

En logique bipolaire (TTL), on utilise la *logique positive*, c'est-à-dire que le 1 logique est représenté par la tension positive la plus élevée.
En TTL, ce sera de 2 à 4 volts, couramment 3 volts, la valeur étant fonction de la charge aval. Le zéro logique, lui, sera représenté par une tension comprise entre 0 et 0,8 volt.

En MOS canal p, la technologie impose une *logique négative...* le zéro logique vaut environ 0 volt, et le un logique la tension la plus négative, environ — 10 volts. Voilà qui promet des acrobaties avec la TTL !

En MOS canal n, enfin, on utilise comme vu une logique positive : le zéro logique vaut toujours environ 0 volt, et le un logique + 2 v à 3 v. Là, nous sommes presque sauvés. La compatibilité n'est cependant pas totale avec la TTL, mais il suffira simplement en sorties de cette dernière d'insérer des résistances dites de *pull-up* qui remonteront le un logique du minimum de 2 v garanti jusqu'aux 3 v désirés : la compatibilité est dès lors assurée, mais au prix de l'utilisation de composants discrets sur nos circuits imprimés.

Notons enfin qu'une faible consommation (MOS par exemple) reste quand même un atout intéressant — non pas pour économiser l'énergie mise en jeu qui est de toute façon bien faible ! — mais pour diminuer l'échauffement de l'ensemble de l'électronique : on gagne ainsi nettement en fiabilité.

2 - 6 - 4. Sournoises difficultés

Malgré tous les efforts, l'idéal «famille» n'est qu'imparfaitement atteint, et nous aurons quand même à nous occuper un peu d'électronique. Citons quelques exemples pour clore ces paragraphes :

— Par-ci par-là, il nous faudra disposer quelques condensateurs pour régulariser les «coups de bélier électriques» dus à de brusques appels de courant, et qui risquent de se propager au long des circuits en entraînant une instabilité dangereuse : ce sont les *capacités de découplage,* une capa de quelques nanofarads tous les 3 ou 4 DIP sera la bienvenue.

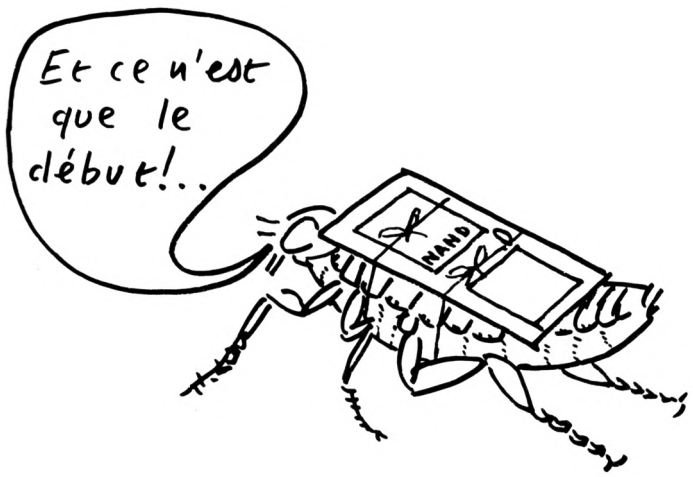
— Les longueurs de fil en logique rapide. La réalisation de l'horloge de Nanostar (pour réduire le prix, nous n'avons pas voulu utiliser

une horloge à quartz du commerce, trop coûteuse) nous a donné du fil à retordre... Disons sans autres commentaires que pour obtenir les beaux créneaux rectangulaires à 20 MHz nécessaires pour piloter tout le fonctionnement du micro-ordinateur, il faut passer convenablement le cinquième harmonique pour avoir des fronts assez raides, et ceci représente cent mégahertz...

— Les problèmes de masse, surtout en logique rapide. Ce n'est pas parce qu'un chemin métallique quelconque relie tous les points à la masse que les problèmes sont entièrement résolus : plus la fréquence est élevée, plus le courant se localise près de la surface du conducteur (c'est *l'effet de peau*) et c'est finalement non pas tant la section que la surface du conducteur qui entre en jeu !

CHAPITRE 3
μP : LE MICROPROCESSEUR

On l'a vu à travers les apparitions successives des diverses technologies — et au sein de chacune d'entre elles par perfectionnement progressif — tous les efforts ont tendu vers un but essentiel : diminuer la taille des composants (et par là même leur consommation) pour augmenter leur nombre sur une même surface de silicium. Autrement dit, augmenter l'intégration.



Au tout début, on a réussi à mettre une porte complète sur une puce (c'était déjà beaucoup lorsqu'on comparait à l'encombrement des cinq ou dix composants discrets nécessaires, surtout une fois câblés...). Puis on en a mis deux, quatre, puis un additionneur, et petit à petit on a franchi des paliers d'intégrations de plus en plus élevés :

— intégration faible (SSI, *small scale integration*),
— intégration moyenne (MSI, *medium scale integration*), avec plusieurs dizaines de portes par puce,
— intégration forte (LSI, *large scale integration*), plusieurs centaines de portes par puce...

... et le vieux rêve est devenu réalité. Rêve de toujours, d'ailleurs : du temps des premiers récepteurs radio à ondes courtes, un amateur s'était mis en tête de rassembler tous les éléments du poste à l'intérieur d'une seule et unique lampe, tout sous vide. Ici, ce sont les concepteurs de circuits intégrés qui ont œuvré continûment pour rassembler sur une même microplaquette suffisamment de composants pour atteindre le seuil décisif : le microprocesseur.

3 - 1. QU'EST-CE QU'UN MICROPROCESSEUR ?

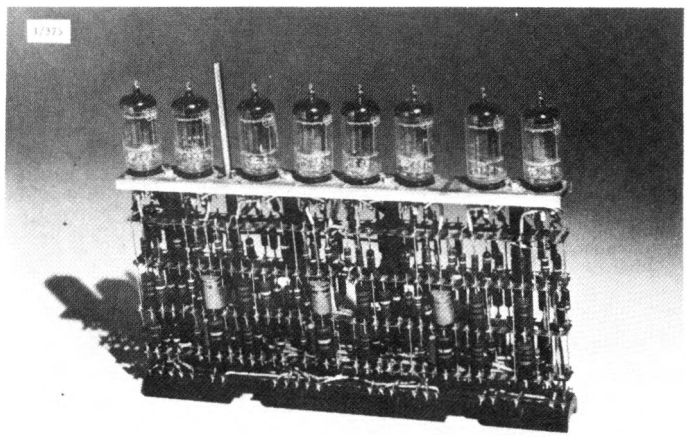
Le voici donc — tout juste sorti du laboratoire — ce fameux microprocesseur qui va bousculer toute l'industrie de l'informatique et nos professions !

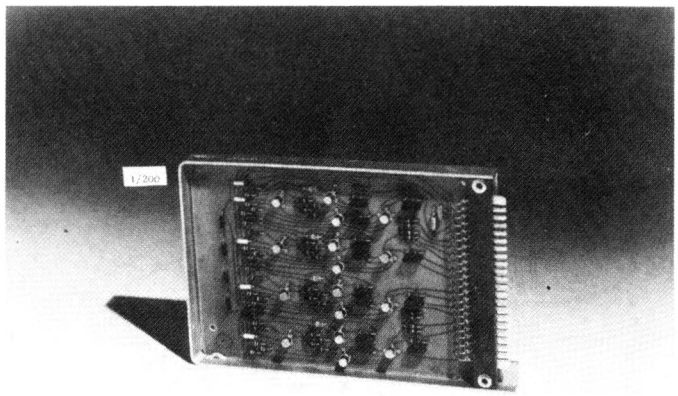
Ce n'est pas encore l'ordinateur complet, mais peu s'en faut. Encore que des machines de poche, dont la Hewlett-Packard HP-65 a été la première représentante, possèdent toutes les caractéristiques d'un ordinateur série simplifié travaillant bit par bit, mais leurs unités centrales n'ont pas été mises sur le marché «au détail», nous verrons plus loin pour quelles raisons.

3 - 1 - 1. Sur une pastille de silicium

Qu'est-ce donc qu'un microprocesseur ? C'est un *processeur* au sens gros ordinateur (y compris la partie commande qui reconnaît les instructions et coordonne leur exécution) tenant sur une ou plusieurs pastilles. Il ne lui manque donc que la mémoire pour devenir une unité centrale complète. Dans un microprocesseur nous trouvons les éléments suivants :

— un ou plusieurs *registres* d'usage général (type *scratch pad,* bloc brouillon),
— un accumulateur et l'Unité Arithmétique et Logique qui lui est associé (ALU),
— un *compteur ordinal* (ou *registre programme,* ou *registre adresse instruction,* voir *Un Fil d'Ariane* dernière édition d'urgence si ces termes vous laissent rêveur, car vous n'êtes pas alors au bout de vos peines...). Lui seront bien souvent associés des *registres de sauvegarde d'adresse de retour,* formant en général une *pile* qui selon le cas sera sur la pastille elle-même ou en mémoire à l'extérieur. Bien sûr, ces registres servent à revenir des *sous-programmes* de façon automatique,
— un bel écheveau de logique combinatoire qui décode les instructions et gouverne le détail de leur exécution, ainsi que ce qui se passe dans l'ALU et dans les registres généraux,





Un raccourci en images de l'évolution technologique sur 20 ans est présenté par ces cinq modules logiques, tous à la même échelle environ (1/2 grandeur). La fraction du système représentée par un module figure à côté de lui.

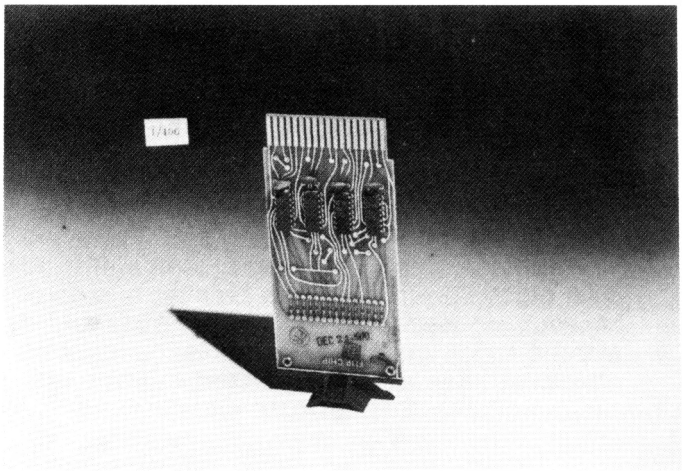
Celui à tubes (vers 1953) était donc accompagné de près de 400 de ses frères dans une machine moyenne de l'époque. Ensuite, ce furent les circuits imprimés et les semiconducteurs. Les quatre clichés qui suivent viennent de Digital Equipment Corporation.

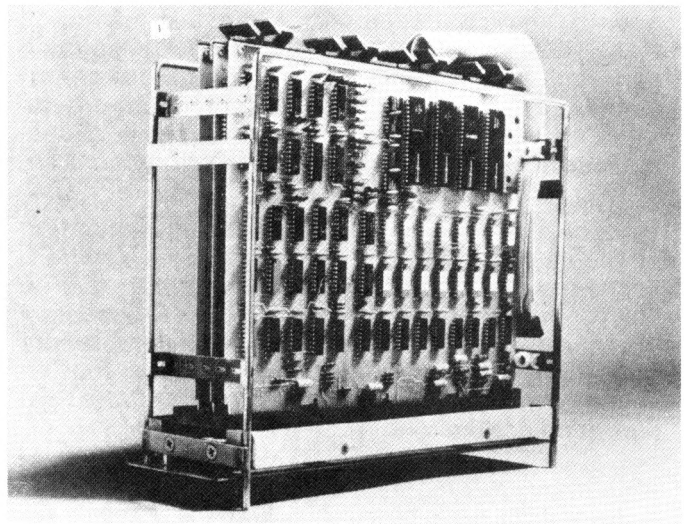
Le premier est un module du premier mini, le PDP-5 (1963) : on n'y voit que des composants discrets, 16 transistors encapsulés individuellement et une centaine de résistances et condensateurs ; tout le câblage visible par transparence à travers le verre-résine est sur l'autre face.

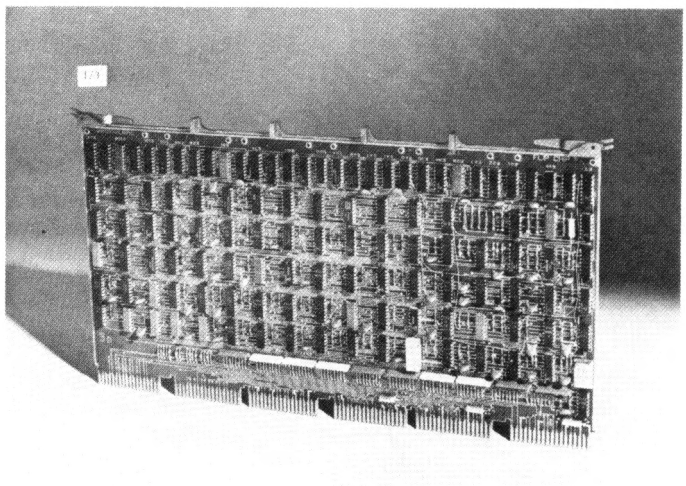
Le second appartient au PDP-8/1 (1968), le I pour Integrated circuits : *ce fut la bombe de l'époque, avec une dizaine de transistors dans chaque DIP à 14 broches. Cette fois, l'augmentation de densité des interconnexions a provoqué l'apparition du circuit imprimé double face ; les rubans du câblage communiquent d'un côté à l'autre par des trous métallisés.*

Le troisième représente le tiers du PDP-8/A (1974) : il s'agit donc toujours de la même machine au plan logique, toujours avec ses mots de 12 bits et des instructions identiques, mais ce n'est plus la même quand on la regarde... ou quand on acquitte la facture. Cette fois, nous avons sur un même circuit imprimé 115 DIPs à 16 broches, contenant chacun de 25 à 50 transistors.

Enfin, le dernier module à trois cartes (en haut à droite) représente le module de base du LSI-11, avec 64 K bits de mémoire (largement extensible). Doté du jeu d'instructions du PDP-11, il est construit autour d'un uP à 16 bits (microprogrammable, en trnches, voir chapitre 3-2) le MPS-1600, fabriqué spécialement pour Digital Equipment par Western Digital. Le module comprend plus de 110.000 transistors, est vendu 1.000 dollars, et est plus puissant que le PDP-5 dont le prix était de 27.000 dollars.







— l'unité de *synchronisation (timing unit)* chargée d'engendrer l'ensemble d'impulsions qui orchestre ouverture et fermeture des portes du réseau précédent ; située sur le chip, elle utilise elle-même les signaux rythmés d'une horloge extérieure,

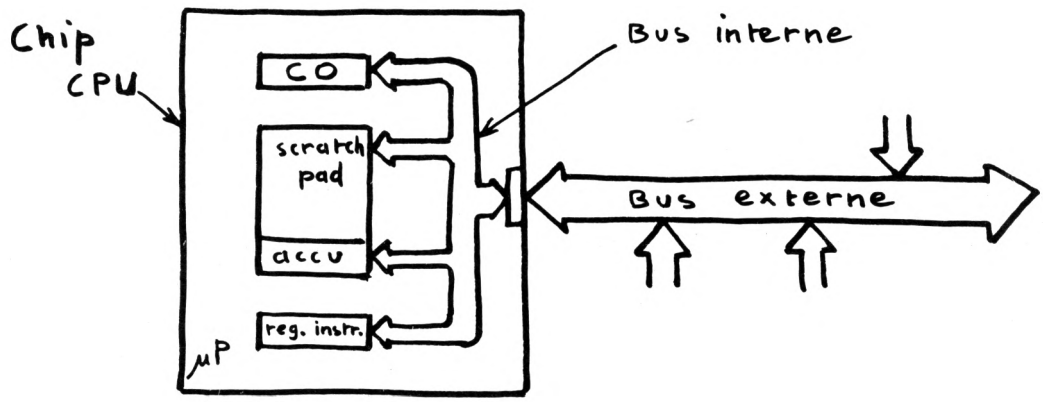
— le *bus interne* du microprocesseur, voie de communication qui permet la circulation de l'information entre les divers éléments que nous venons de citer, et également le suivant,

— un moyen de communication avec l'extérieur, généralement constitué par l'extrémité du bus interne, ou par un *registre tampon* alimenté par le bus.

Et tout ceci tient sur un chip de quelques millimètres carrés ! Après lui avoir greffé une horloge, nous n'aurons qu'à lui rajouter un *bus externe* et y connecter mémoires et périphériques pour obtenir un micro-ordinateur complet. Si l'on songe qu'il existe maintenant des chips mémoire de 4K bits, et très bientôt de 16K, on a une idée visuelle immédiate de l'encombrement total du système... Nanostar dans une toute petite valise.

3 - 1 - 2. Le bus : une autoroute à voies multiples parfois encombrée

Les éléments listés plus haut sont pour la plupart familiers aux informaticiens (ou nous n'avons pas à entrer dans leurs détails, comme



pour le timing par exemple) ; quant aux mémoires RAM, ROM et autres, ainsi que l'horloge, nous en parlerons plus loin. Par contre, le bus qui est un élément essentiel en conception de micro-ordinateurs est nettement moins connu, aussi allons-nous commencer à débroussailler un peu son rôle et sa structure.

Le bus externe est la voie de communication privilégiée du chip microprocesseur : il servira à l'échange d'informations avec l'extérieur, que ce soient des données ou des adresses, les deux grandes catégories d'information. Il comprendra également toujours (intégrées avec lui ou en accompagnement) des *lignes de contrôle* qui auront un double rôle :

— indiquer la présence ou la non-présence d'information sur le bus,
— indiquer la nature de cette information.

Selon le type de microprocesseur nous trouverons des bus dotés de plus ou moins de fils, en fonction en particulier du nombre total de broches du boîtier du microprocesseur, déterminant le nombre de broches qu'on peut envisager de réserver au bus lors de la conception du chip. Citons deux exemples opposés :

— le bus unique du 8008. Lorsque les designers de ce microprocesseur de Intel (qui fut le premier 8 bits mis sur le marché, et connut un succès instantané partout : en France, il est la base du Micral de R2E) se trouvèrent confrontés au problème de relier un DIP à 18 broches au reste du monde, ils ne purent attribuer que 8 broches au bus. On trouvera donc bourre et balle sur ce dernier, selon l'instant :

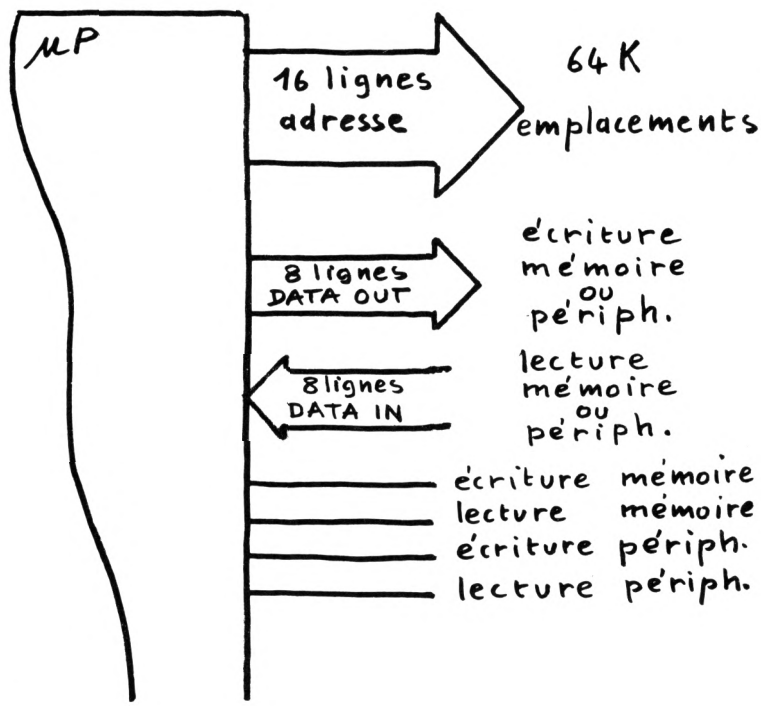
  • des adresses mémoires,
  • des données pour la mémoire ou en provenance de la mémoire,
  • des adresses de périphériques,
  • des données pour les périphériques ou en provenance des périphériques,
  • des informations de contrôle complémentaires.

  C'est la raison pour laquelle un tel bus est dit *multiplexé* : à l'utilisateur de réaliser en TTL le multiplexeur extérieur pour débrouiller tout ceci. Heureusement, nous disposons également de lignes de contrôle au nombre de trois, indiquant à chaque instant l'état du 8008 parmi les huit états possibles : cette information est donc codée en binaire et demande à être décodée extérieurement par une logique en boîtiers TTL — ce qui ne serait pas le cas si huit lignes d'état séparées avaient été possibles. La contrepartie de tout ceci... c'est que le 8008 a été le premier, et reste actuellement remarquablement économique compte tenu de ses possibilités.

— le bus en deux parties du PPS-4 de Rockwell ou du 8080 d'Intel, grand frère du 8008 par ses possibilités... mais en même temps fils du 8008 puisqu'il représente le premier microprocesseur de seconde génération mis sur le marché ! La généalogie œdipienne de nos puces, reflétée par leur numéro de type, est assez piquante. Ce bus comprend donc :

  • une partie réservée aux adresses (mémoires ou périphériques),
  • une partie réservée aux données (mémoires ou périphériques).

Nous en terminerons provisoirement avec le bus, après avoir donné ces quelques notions de base (nous reviendrons ultérieurement sur ce problème du bus pour le définir très complètement) en indiquant dans notre croquis le schéma type d'un bus complet pour un microprocesseur à huit bits permettant d'adresser directement 64K positions, mémoires ou périphériques. Nous voyons qu'il comprend la bagatelle de 36 fils !



3 - 2. BREVE HISTOIRE ET TENTATIVE IMPOSSIBLE (MAIS PLEINE D'ENSEIGNEMENTS) DE CLASSEMENT DES MICROPROCESSEURS

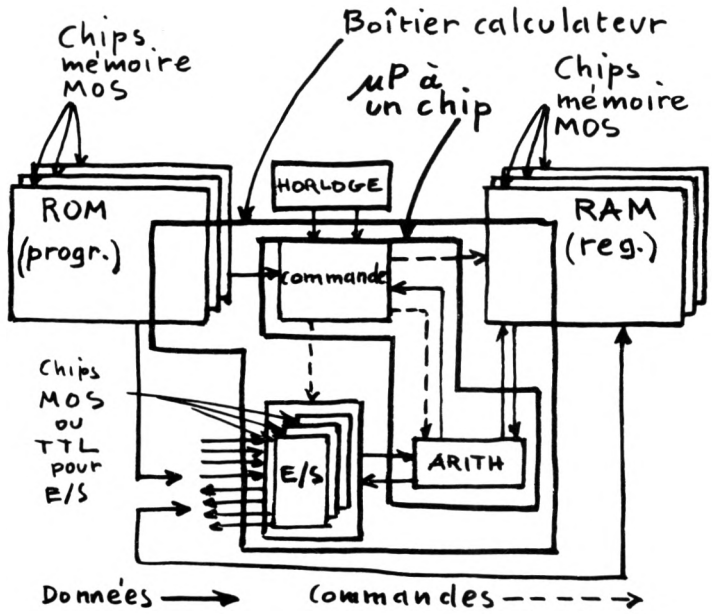
Il est possible de considérer le microprocesseur sous deux éclairages différents — mais complémentaires. Pour le fabricant de composants, il est l'évolution directe des ensembles à deux chips utilisés dans les calculatrices de poches modernes, évolution assurée par les progrès technologiques constants des processus de fabrication que nous avons étudiés. Pour le concepteur logique, il découle, par simplification, de l'architecture évoluée des mini-ordinateurs qui l'ont précédé.

3 - 2 - 1. Les pionniers, quatre à quatre

Essayons donc de suivre les étapes de la brève histoire du microprocesseur, pour essayer d'y voir un peu plus clair dans le fouillis désorientant de chips de toute nature actuellement disponibles sur le marché.

Tout d'abord sont apparus des *boîtiers calculateurs* destinés aux premières calculatrices de poche. Fortement optimisés dans leur conception pour cette tâche précise, et nullement destinés à un usage général, ces boîtiers n'étaient ni programmables par l'utilisateur, ni par conséquent destinés à recevoir des programmes enregistrés destinés à des applications variables. La programmation des routines de la calculatrice (balayage du clavier pour entrée des données, rafraîchissement de la visualisation pour affichage des résultats, calculs, etc.) était (est encore) entièrement figée en mémoire *ROM* où les instructions sont photogravées en cours de fabrication au moment de l'attaque de la métallisation (nous reviendrons en détail sur cette importante question des mémoires, les *RAM* étant, elles, identiques à celles que nous connaissons sur les gros ordinateurs : on peut les lire *et* les écrire, alors qu'on ne peut *que* lire les ROM, bien évidemment). L'unité arithmétique comportait naturellement les quatre fonctions, obtenues à partir d'une (l'addition) et de la complémentation, séquencées par les ordres en ROM.

Dans une recherche de diminution extrême du coût, et ceci était possible dans le cadre de la technologie vu la complexité moyenne de l'ensemble, processeur et mémoire sont en général réunis sur une même puce, donc dans un même boîtier. Seule l'horloge, fabriquée avec une technologie différente, reste en dehors. Mi 1973, le TMS-0117 de Texas Instruments, version évoluée lisant des mots codés de 5 bits pour chaque pression de touche, avec une sortie DCB de 4 bits plus trois bits d'état, valait par lot de mille... 10,58 dollars pièce !



Evolution du boîtier calculateur vers le calculateur à 2 chips {CPU + ROM/RAM et le µP, fort voisins.

Puis des exigences supplémentaires quant aux calculatrices sont apparues : les rendre programmables, et/ou imprimantes. La solution était évidemment de séparer cette fois l'affaire en deux chips:

— la puce du microprocesseur, où débarassé de l'espace occupé par la ROM, il a été possible d'étendre les possibilités en registres et en instructions,
— la puce ROM où, de même, on pouvait loger des programmes plus importants.

Cette fois, nous tenons notre ensemble à deux chips permettant des calculatrices de pointe. Naturellement, ayant maintenant les fonctions arithmétiques et commande nettement séparées du reste, il est possible d'adjoindre à cet ensemble, devenu modulaire, d'autres boîtiers de mémoires ROM, RAM, d'interfaces d'entrées-sorties variées, etc.

C'est Intel qui a mis le premier sur le marché en décembre 1971 un chip de ce type, le 4004, qui accompagné des autres chips membres de sa famille est dénommé MCS-4. Comme le chip microprocesseur (ou μP) tout seul est de peu d'utilité, sans son programme, on parle donc surtout de MCS-4. Rockwell avec le PPS-4 et Fairchild avec le PPS-25, toujours dans la même catégorie, suivirent bientôt.

Ces machines sont assez déroutantes de prime abord : leur architecture paraît très obscure. Une machine comme le MCS-4, par exemple, n'a pas de registre adresse mémoire ! A première vue, on se demande même comment on peut l'utiliser. Quant au PPS-4, certaines de ses instructions sont si bizarres dans leur fonctionnement qu'on mordille nerveusement son crayon à la lecture de leur description. Jugez en plutôt d'après celle-ci : elle échange le contenu de l'accumulateur avec celui d'un mot situé en mémoire RAM, effectue un OU exclusif sur les trois derniers bits de l'adresse RAM, décrémente de un le registre adresse RAM, teste le franchissement de 1111, et le cas échéant saute l'instruction suivante...

3 - 2 - 2. Architectures Harvard et Princeton

L'origine de bon nombre de difficultés — et d'étrangetés — que nous rencontrons ici tient au fait que ces microprocesseurs, au contraire des ordinateurs classiques, ne sont pas tout à fait des machines de Von Neumamm : ils manipulent souvent les mots instructions séparément et différemment des mots de données.

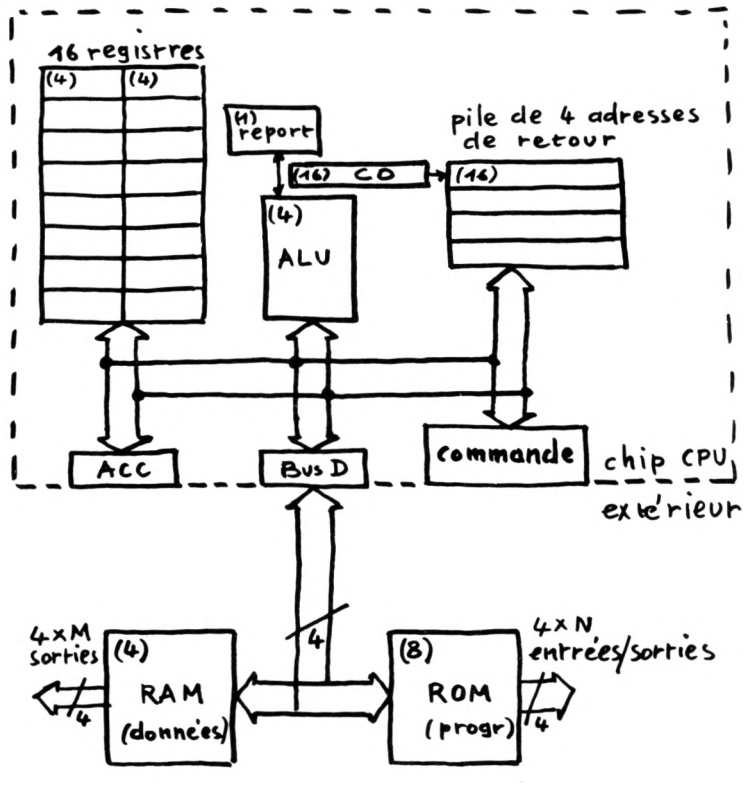
Voici donc violé le grand principe, celui qui avait permis de passer de l'architecture «Harvard» des premières machines (où la mémoire centrale ne servait qu'aux données et résultats de calcul, le programme étant enfiché à la demande sur un tableau comme pour une tabulatrice ou un calculateur analogique) à l'architecture «Princeton» où, selon la géniale idée du grand John, une mémoire centrale étendue, et dont tous les mots étaient *banalisés*, servait en plus à *enregistrer le programme*.

Dans une machine à l'architecture Harvard, programme et données sont par conséquent stockés de façon différente, et ne peuvent être mélangés : il n'est donc pas possible d'utiliser directement des données comme instructions, puisqu'elles sont dans des mémoires séparées, ni de modifier des instructions dans l'unité arithmétique et logique (sauf en utilisant l'adressage indirect). Bien entendu, dans les systèmes spécialisés où les microprocesseurs ont été utilisés en majorité, comme de toute façon le programme est prévu pour être figé en mémoire morte ROM, ceci ne constitue vraiment une limitation gênante que pendant la conception et la mise au point des programmes.

A quel point la mémoire ROM dans ces machines est la mémoire principale, est attesté par le fait que le compteur ordinal contrôle l'accès à la mémoire ROM, et que les instructions de saut ont effet sur les adresses ROM. La mémoire RAM est pratiquement réduite au rôle de périphérique !

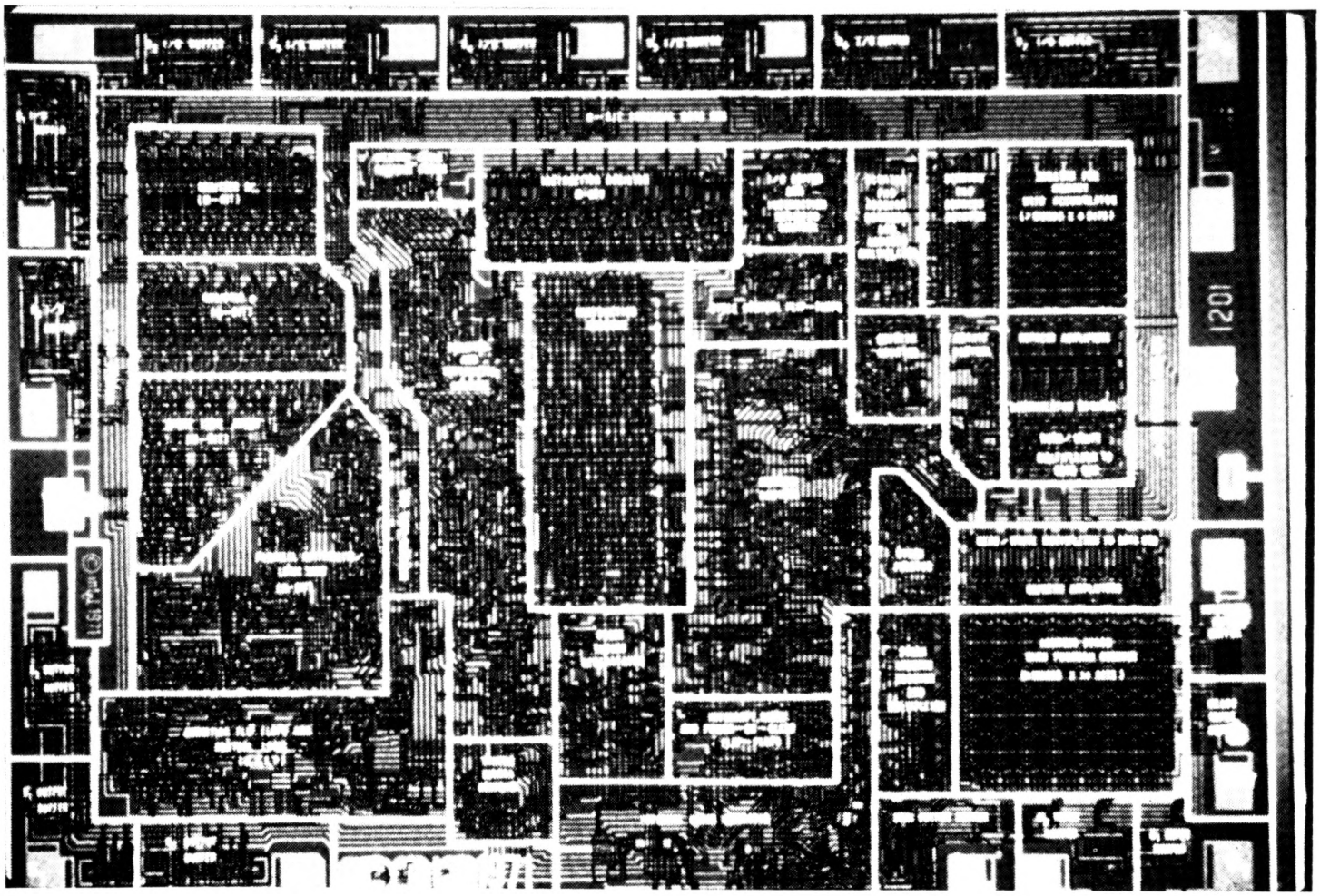
Nous avons figuré par un croquis l'architecture du MCS-4. On remarque que le processeur est abondamment pourvu en registres généraux de 4 bits, au nombre de 16, en plus de l'accumulateur, de sorte que pour de nombreuses utilisations spécialisées le concepteur découvre qu'il peut complètement se passer de RAM externe, vu la capacité de scratch-pad dont il dispose sur la puce elle-même.

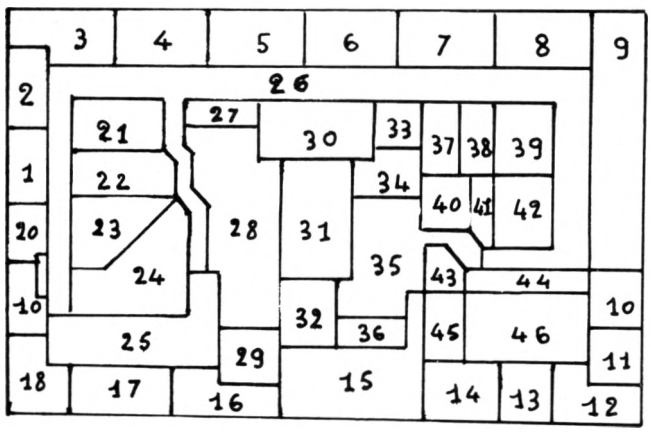
Donnons quelques derniers détails bien «typés microprocesseur». Le lecteur a déjà sans doute réagi avec surprise en découvrant ce qui paraît à première vue une erreur : sur la RAM on a marqué une sortie seulement... et sur la ROM une entrée/sortie ! En fait, ces chips mémoire jouent un rôle multiple. On constate que ce n'est *qu'à eux* qu'est relié le bus : il ne peut être connecté directement à aucun périphérique.



Architecture "Harvard" du MCS-4

Chaque puce mémoire (ce sont des boîtiers spécialisés par conséquent pour être utilisés avec le 4004) comporte par contre un registre de 4 bits, relié entre autre au bus, et prévu pour connexion sur les périphériques. Sur les RAM il ne peut servir qu'à la sortie, et sur les ROM, par masquage à la fabrication, on choisit entrée *ou* sortie. Selon l'application on choisira donc le nombre et la nature de boîtiers mémoire en fonction des entrées/sorties, aussi bien qu'en fonction des capacités mémoires nécessaires.





Le chip du μP 8008 vu au μscope de poche.

1) D_0 — Buffer d'E/S et plot de connexion
2) D_1 — id.
3) D_2 — id.
4) D_3 — id.
5) D_4 — id.
6) D_5 — id.
7) D_6 — Buffer d'E/S
8) D_7 — id.
9) Plots de D_6 et D_7, alimentation électrique V_{DD}
10) INT — Buffer d'entrée de l'interruption
11) RDY — Buffer d'entrée du signal ready
12) Buffer d'entrée de Φ 1
13) Buffer d'entrée de Φ 2
14) Buffer de sortie de SYNC
15) Générateur d'horloge interne
16) Un coup pour rien
17) Buffer de sortie de S_0
18) id. pour S_1
19) id. pour S_2
20) Alimentation électrique V_{CC}
21) Registre temporaire b (8 bits)
22) Registre temporaire c (8 bits)
23) Circuits d'anticipation de report sur 8 bits (voir les différents articles *addition* dans l'Encyclopédie, *l'Informatique nouvelle*, 1974, Editions d'Informatique)
24) Unité arithmétique et logique parallèle sur 8 bits
25) Flip-flops de condition (C, Z, S, P) et logique de contrôle
26) Bus interne (largeur 8 bits)
27) Circuits cycle mémoire
28) Commande registre et arithmétique
29) Décodage de signaux d'état
30) Registre instruction (8 bits)
31) Décodeur d'instructions
32) Générateur des signaux temporels d'état
33) Gestion du registre instruction et des buffers E/S
34) Flip-flops de commande de cycle
35) Commande des mémoires internes
36) Flip-flops de remise à zéro à la mise sous tension, de Ready et de Interrupt
37) Multiplexeur d'adresses et pointeur d'adresses des registres généraux
38) Décodeur d'adresse des registres généraux
39) Registres généraux et accumulateur (7 mots de 8 bits)
40) Compteur de rafraîchissement
41) Compteur de décalages
42) Amplificateurs de rafraîchissement, circuits de décalage de l'accumulateur, multiplexeur de lecture/écriture relié au bus interne
43) Pointeur de pile
44) Multiplexeur de lecture/écriture relié au bus interne et amplificateurs de rafraîchissement
45) Décodeur et multiplexeur de pile d'adresses
46) Pile d'adresses et compteur ordinal (8 mots de 14 bits)

Le PPS-4, lui, est un peu plus ouvert vers la mémoire vive RAM, puisqu'il possède en plus un registre adresse spécialisé servant aux accès en RAM, d'un rôle proche de celui du compteur ordinal, et comportant ses propres instructions spécialisées. Nous reparlerons plus tard en détail du PPS-4 que nous avons retenu comme 4 bits typique à famille de chips particulièrement intéressante.

En conclusion, ces microprocesseurs à 4 bits, extrêmement bon marché, bien qu'assez fortement typés sont loin d'être dénués d'intérêt. Leur destination initiale les rend particulièrement performants pour la manipulation arithmétique de chaînes de caractères décimaux de 4 bits. Le MCS-4, le plus lent, exécute quand même une instruction en 10,8 à 21,6 μs, le PPS-4 en 5 à 10 μs. Le Fairchild PPS-25 paraît plus lent puisqu'il prend toujours 62,5 μs pour un cycle, mais dans ce temps il traite non pas 4 bits comme les autres, mais une chaîne de 25 caractères DCB, ce qui le rend le plus rapide ! La taille maximale du programme en ROM, limitée par l'adressage, est de 4.096 instructions de 8 bits pour le MCS-4, de 6.656 de 12 bits pour le PPS-25 et de 16.384 de 8 bits pour le PPS-4, tout ceci étant déjà assez respectable.

Et le 4004 existe maintenant dans une version récente beaucoup plus puissante et rapide, le 4040...

3 - 2 - 3. Le 8 bits entre en scène

Le véritable coup d'envoi des microprocesseurs a été donné par l'apparition en avril 1972 du premier 8 bits, le 8008, capable d'exécuter une instruction sur 8 bits en registres généraux en 20 μs... et cette fois encore réalisé par Intel, comme nous l'avons vu, le système étant dénommé MCS-8. La popularité de ce microprocesseur a été immédiate, une longueur de mot de 8 bits offrant d'incontestables avantages : le meilleur compromis entre les possibilités de traitement de la technologie MOS/LSI d'une part, et les besoins informatiques d'autre part dans les domaines communications, contrôle de processus et traitement de texte en particulier. Les caractères à huit bits ASCII ou EBCDIC sont devenus la norme, deux caractères DCB peuvent être traités à la fois, et 8 bits en simple ou double longueur conviennent bien au domaine industriel.

Les concepteurs de ce huit bits première génération travaillaient avec de dures contraintes. Le design de la machine (le cycle de développement d'un μP est de plusieurs années) était en cours en 1971, et pour des raisons de coût de revient/prix de vente, un DIP à 18 broches seulement fut imposé. D'autre part, la taille du chip réalisé en MOS canal-p, devait être la plus petite possible, de façon à obtenir comme nous l'avons vu de bons rendements de fabrication, toujours dans le même but. De fait, la puce de 8008 mesura seulement 3 mm sur 4,3 mm. Tout ceci avait forcé le designer Hal Feeney à s'écarter notablement de l'architecture d'un ordinateur classique, au-delà même de ce qu'imposait déjà la taille de mot très courte : le décorticage du bus du 8008 nous a donné un échantillon des «contorsions» nécessaires... nous empruntons ce terme à un concurrent d'Intel ! Nous n'en dirons pas plus pour le moment, le MCS-8 (avec en particulier la version 8008-1 qui est passée de 20 à 12,5 μs) étant suffisamment important pour que nous lui consacrions plus loin un développement spécial.

3 - 2 - 4. Les électrons dépassent les trous, et le 8080 dépasse le 8008

Le progrès technologique poursuivant son rythme — soutenu s'il en fût — à peine avions-nous pris conscience de l'existence du premier huit bits et de ses potentialités que, vers fin 1973, les revendeurs de composants électroniques alignaient sur leur étagères une nouvelle merveille : le premier 8 bits de seconde génération en MOS canal-n... toujours par Intel qui n'avait pas lâché le flambeau. L'année 1974 a donc été l'année de décollage du μP à 8 bits.

Le 8080 présente une différence importante immédiatement visible : le chip est conditionné cette fois dans un boîtier DIP à 40 broches. Il a ainsi été possible à Federico Faggin, qui dirigeait l'équipe des trois concepteurs du 8080, d'adopter le bus de grande largeur dont nous avons déjà donné la description, supprimant de cette façon le multiplexage nécessaire sur le bus étroit du 8008 : outre une simplicité d'utilisation nettement meilleure, entraînant d'ailleurs une notable économie en boîtiers TTL externes, le fait que le μP n'a plus besoin de partager dans le temps l'utilisation d'un petit nombre de broches, embouteillées par des informations de natures différentes, lui permet d'exécuter ses instructions plus rapidement. Ajoutons à ceci l'accroissement de mobilité des électrons par rapport aux trous, il s'ensuit que les instructions qui s'exécutaient en 20 μs sur le 8008 s'exécutent maintenant en 3 μs !

On le voit, le saut est de taille, un 8 bits seconde génération commence à mordre sérieusement dans le territoire des minis : nous tenons là des microprocesseurs à prendre très au sérieux, dont les applications possibles sont vastes, avec un adressage direct jusqu'à 64K octets. Ceux qui n'ont pas la mémoire trop courte se rappelleront sans doute que les 360/30 ont souvent été commandés dans les débuts avec 32K octets, pour passer après un an ou deux à 64K lorsque les applications se développaient...

3 - 2 - 5. Emulation sur les 8 bits canal-n

Environ un an plus tard commencèrent à apparaître d'autres 8 bits de seconde génération, en particulier le Motorola 6800, puis le Signetics PIP, puis bientôt de nombreux autres.

Le 6800 est un μP à la structure intéressante, de vitesse voisine du 8080, doté d'une bonne variété de modes d'adressage, avec un véritable registre d'index de 16 bits. Il a été annoncé d'autre part avec une famille de chips d'accompagnement bien conçus, qui arrivent progressivement les uns après les autres sur le marché. Au sujet de 6800, une anecdote va nous montrer à quel point — fait qu'on a souvent tendance à oublier — le développement des microprocesseurs représente de nombreux mois de labeur et un véritable tour de force aux frontières extrêmes de l'avance technologique. Dans le labyrinthe des dessins de la puce, le problème des erreurs, surtout des petites, est omniprésent, forçant parfois à de douloureux retours en arrière. Ainsi, lorsque l'équipe de Motorola — au-delà de la moitié du travail de design — se rendit compte qu'un bit drapeau de condition était resté en rade quelque part dans le dessin... ils continuèrent pour terminer le premier chip du 6800 tel quel, jugeant que le travail serait quand même fructueux en enseignements, quitte à reprendre plus tard de nouveau le design en rajoutant le bit manquant.

Le problème de la sélection d'un microprocesseur n'est donc pas une affaire à prendre à la légère. On investit beaucoup de temps à approfondir les détails d'utilisation hard et soft d'un chip, et il vaut mieux bien assurer ses arrières pour ne pas avoir aussi à faire demi-tour en chemin, en tant qu'utilisateur cette fois. Le principal mérite d'un microprocesseur est d'abord d'*exister*, nous voulons dire par là non pas d'être livrable à l'échantillon, rescapé de la masse de rebuts trahissant un rendement de fabrication infime, mais d'être disponible dans les casiers des détaillants. Attention donc, de nombreux microprocesseurs sont, pendant de longs délais, des tigres de papier (nous ne disons pas cela pour le 6800) les fabricants ayant un peu tendance à forcer leur talent.

Il y a donc maintenant trop de 8 bits pour que nous parlions de chacun d'eux (vous trouverez leurs caractéristiques dans le tableau d'ensemble), le dernier pour lequel nous le ferons sera le PIP. Venant assez tard, les techniciens de Signetics ont cherché pour leur *Programmable Integrated Processor* à jouer la carte de la simplicité d'utilisation. Voisin du 8080 et du 6800, le PIP présente des caractéristiques assez rares : il est statique, permet 8 modes d'adressage et est muni d'un mot d'état de programme. Voyons donc ces points d'un peu plus près, ils sont instructifs :

— statique : en effet, la plupart des autres chips sont dynamiques, c'est-à-dire que les mémoires utilisées (registre, accu, etc.) s'évanouissent si l'on essaye de faire fonctionner le microprocesseur avec une vitesse d'horloge trop faible (nous en verrons la raison dans le sous-chapitre sur les RAM dynamiques). En particulier, impossible de faire du pas à pas sans adjonctions spéciales, ce qui ne facilite évidemment pas les mises au point. Ce problème est éliminé avec le PIP.

— 8 modes d'adressage. C'est à peu près le maximum imaginable, donc souplesse totale de côté. Par contre, l'adresse n'a que 15 bits (32K),

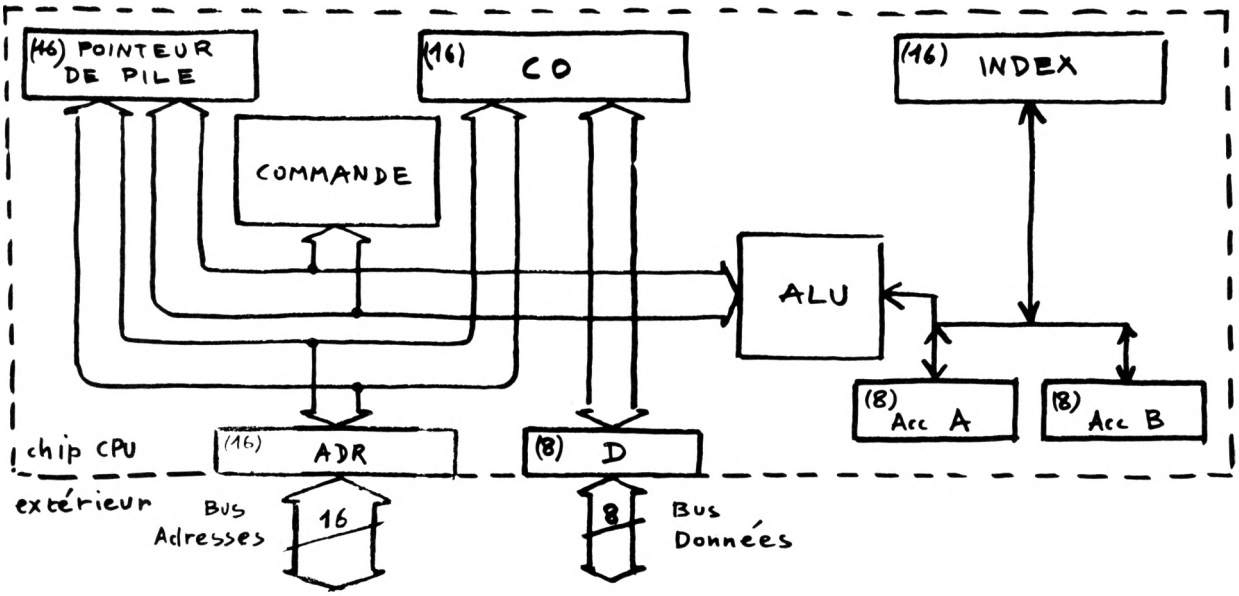
— mot d'état : le PSW (*Program Status World*), qu'on a vu apparaître sur le 360 IBM, est un regroupement de tous les bits de conditions, flags, états du processeur, interruptions, etc., normalement éparpillés un peu partout dans le processeur, en un seul mot. Il est donc ainsi facilement possible de sauvegarder ou de modifier le PSW. Il comprend ici 14 bits utiles dans 2 octets :

- pointeur de 3 bits pour indiquer le registre où se trouve l'adresse du retour,
- deux bits indiquant l'état de l'unité arithmétique et logique après une opération,
- un bit pour changer le signe d'une retenue, selon qu'il s'agit d'une addition ou d'une soustraction,
- un bit indicateur de dépassement de capacité,
- un bit pour inhiber les interruptions,
- un bit drapeau à destination de l'extérieur,
- un bit drapeau enregistrant un signal extérieur,
- un bit de retenue interchiffre (pour des opérations décimales),
- deux bits donnant un résultat de comparaison, et enfin,
- un bit pour indiquer quelle moitié des registres généraux est adressée.

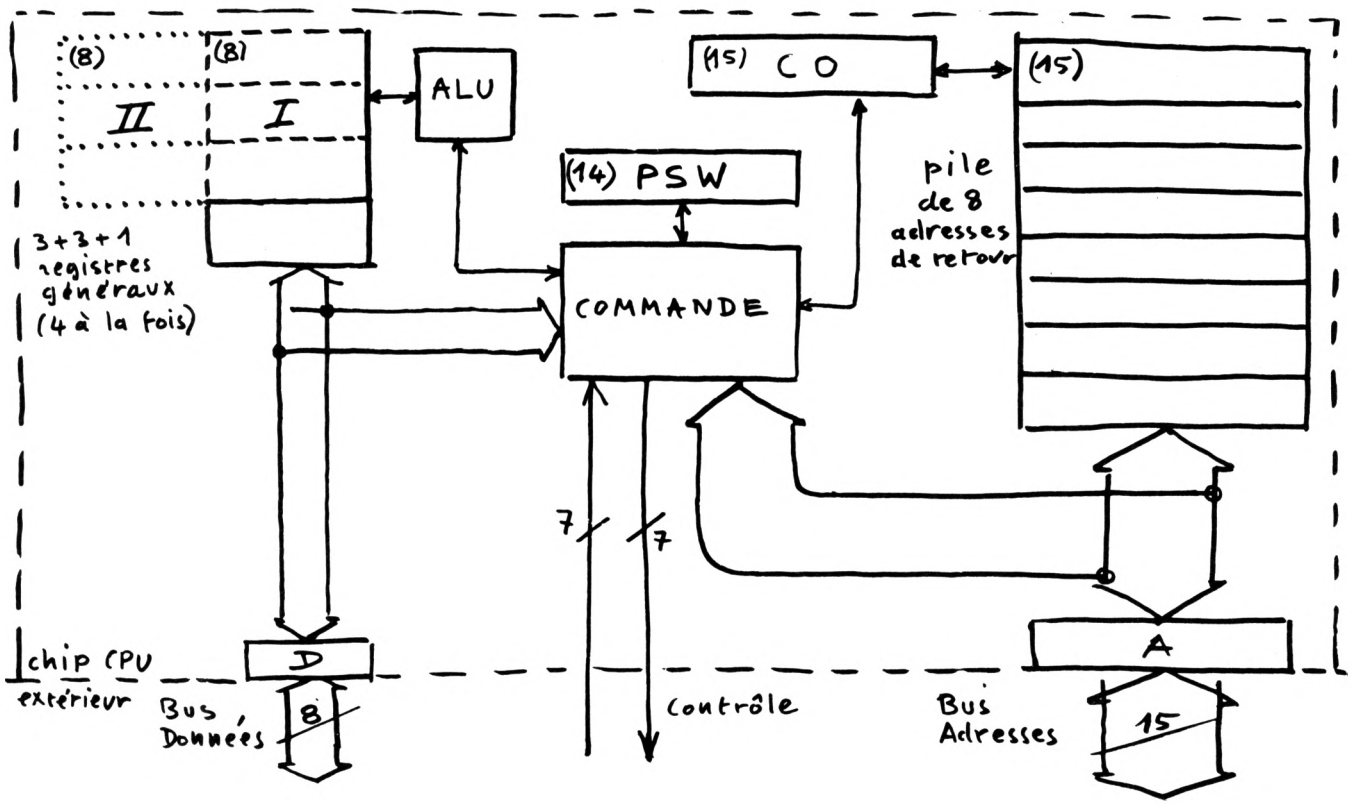
3 - 2 - 6. Microprogrammes et tranches de machines

Tous les chips dont nous avons parlé jusqu'à présent, possèdent «par construction» un jeu d'instructions fixé au départ par leurs designers, c'est-à-dire qu'ils ont un *code d'ordres câblé*. Ce ne sont donc pas des machines *microprogrammées* : le préfixe micro, dans microprocesseur, s'applique uniquement à leur dimension physique millimétrique.

Il existe par contre des microprocesseurs qui — comme la plupart des ordinateurs de troisième génération, selon le principe fécond imaginé par M.V. Wilkes en 1951 — ont un jeu d'instructions

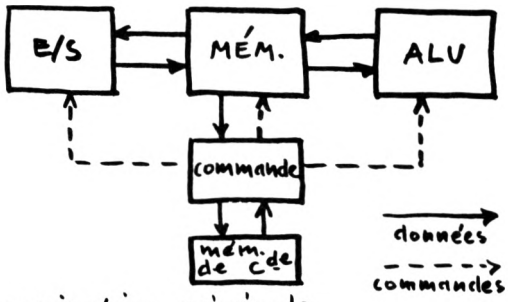


Architecture du Motorola 6800



Le PIP 2650 de Signetics.

Un bit du PSW indique si la série des trois registres généraux en cours est I ou II.

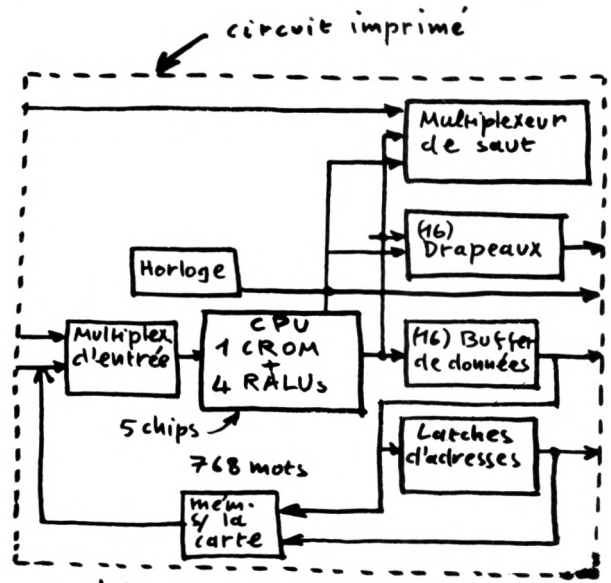


Organisation générale d'une machine microprogrammée

déterminé par des microprogrammes : on pourrait donc dire qu'ils sont doublement micro !

Dans ces μP, on voit donc apparaître un élément nouveau : la *mémoire morte de commande* CROM *(Control read-only memory)*. Notre premier schéma montre sa relation aux circuits qui contrôlent l'unité arithmétique et logique, les lignes continues indiquent les transferts de données, et les lignes en tirets les commandes.

Chaque instruction du programme écrit par l'utilisateur (instruction et programme étant pris ici avec leur signification habituelle, celle qu'ils ont pour les machines à code d'ordres câblé) est en réalité une *adresse* indiquant où se trouve dans la mémoire de commande le début d'une série de micro-instructions à exécuter (une séquence peut ne contenir qu'une micro-instruction). La CROM, comme l'indique le second schéma bloc, comporte donc un registre adresse et un décodeur associé à la ROM. Dans certaines configurations, le registre adresse est automatiquement incrémenté pour passer à la micro-instruction suivante, comme un compteur ordinal d'ordinateur. Cependant, dans la majorité des cas, il n'en est pas ainsi

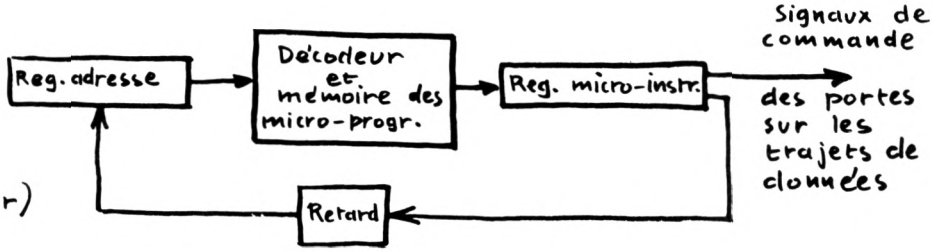


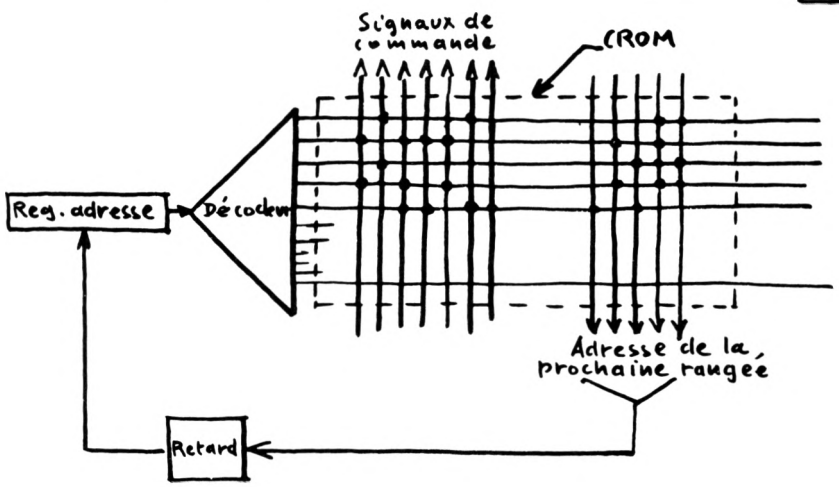
Architecture du National IMP 16C

Le troisième schéma indique en effet de façon un peu plus détaillée la structure de la CROM : on y voit que les micro-instructions ne s'exécutent pas obligatoirement séquentiellement. Il va sans dire qu'il s'agit ici de la programmation la plus fine qui

Schéma simplifié du fonctionnement de la CROM →

(le retard donne le temps à la micro-instruction de s'exécuter)





← Schéma détaillé :

le contenu du registre adresse active une ligne horizontale par l'intermédiaire du décodeur. La ligne contient une micro-instruction : chaque connexion indiquée par un point représente un bit 1.

La partie droite de la micro-instruction contient l'adresse de la suivante.

soit, systématiquement au niveau du bit, chacun d'eux commandant une fonction élémentaire, telle que l'ouverture ou la fermeture d'une série de portes quelque part dans la RALU.

Car bien sûr la CROM est accompagnée d'une unité arithmétique et logique, réduite à sa plus simple expression mais comportant un certain nombre de registres, d'où le R de RALU. La largeur des bus internes et des registres d'un chip RALU est assez faible, typiquement deux ou quatre bits, mais ils sont conçus pour être juxtaposés (électriquement sinon physiquement) de façon à former un processeur ayant la taille de mot désirée : ce sont des processeurs en tranches.

Prenons un exemple précis, celui des chips de National Semiconductor. Quatre DIP RALU de quatre bits, accompagnés de leur CROM, nous permettent d'obtenir un processeur à mots de 16 bits, le IMP 16C. Chaque RALU comprend :

— sept registres de quatre bits,
— une pile de 16 mots,
— quatre indicateurs d'état,
— une ALU de quatre bits capable d'exécuter ADD, AND, OR, EOR et décalage.

La CROM peut accepter de 1 à 8 RALLU (donc on peut assembler une machine à mots de 32 bits) et contient lorsque livrée par NS le microprogramme concrétisant un code d'ordres standards de 43 instructions.

Notons ici au passage que — pour mieux clarifier la terminologie — on peut dans le cas de machines microprogrammées appeler macro-instructions les instructions du programme écrit par l'utilisateur (ce qui par contre risque d'introduire de la confusion si l'on écarte du matériel pour commencer à parler de logiciel, car alors quid des macro-assembleurs et des macro-instructions... il faudrait parler de macro-macro-assembleur et de macro-macro-instructions !).

Mais revenons à notre IMP 16C. La mémoire morte comprend 100 micro-instructions de 23 bits chacune, dont les configurations binaires spécifient pour la RALLU les commandes suivantes :

1 : registre source A,
2 : registre source B,
3 : sélection de l'opération ALU à effectuer,
4 : registre destination R.

Ces commandes correspondent aux quatre phases composant un cycle d'horloge complet dont la durée est de 15 μs. A titre d'indication, la durée moyenne d'exécution d'une macro-instruction est de 7 μs.

On le comprend bien, pour que les macro-instructions puissent se dérouler à une vitesse convenable, il faut que la ROM ait un temps d'accès assez court, permettant l'emploi d'une deuxième horloge disons cinq fois plus rapide que celle utilisée par le reste du système, en particulier la mémoire où est logé le programme d'application (le macro-programme, si l'on veut). L'exemple NS illustre la *microprogrammation polyphase*, dans laquelle les micro-instructions sont suffisamment puissantes pour nécessiter plusieurs impulsions d'horloge pour leur déroulement, ce qui sera plus complexe mais plus rapide. En monophase, il y aura davantage de micro-instructions plus courtes.

Naturellement, l'utilisateur peut ne pas s'en tenir au code d'ordres standard (l'utilisateur très éclairé, du moins), et la micro-programmation lui offre alors le moyen de créer des macro-instructions spéciales capables d'exécuter à très grande vitesse des opérations complexes telles que :

— calcul de fonctions mathématiques, trigonométriques, etc.,
— multiplication et division,
— opérations en virgule flottante,
— opérations matricielles,
— transformée de Fourier rapide,
— commandes de périphériques, etc.,

au prix bien entendu de pas mal de «midnight oil», comme on dit de l'autre côté de l'eau.

Citons comme dernière utilisation de la microprogrammation (il y en a bien d'autres) celle relative aux tests du matériel. Pour s'assurer du bon état du hardware, il faut pouvoir vérifier l'intégralité de tous les trajets que peuvent emprunter les données, tâche difficile lorsqu'on travaille au niveau du langage machine. Par contre, des routines de micro-diagnostic peuvent vérifier individuellement chaque transfert, décelant les défaillances au niveau optimal pour le technicien d'entretien, puisque les micro-instructions permettent de tester des fonctions simples au lieu de combinaisons de fonctions comme dans le cas des tests software traditionnels. Au surplus, les micro-diagnostics sont plus rapides et utilisent moins de place en mémoire.

3 - 2 - 7. Tentative de classification

Et finalement nous voici au pied du mur. Comment essayer de classer ce fouillis de microprocesseurs, pour y voir un peu plus clair ?

Les critères auxquels on peut faire appel sont nombreux, et bien sûr le premier est la longueur du mot, le plus utilisé dans le cas des machines traditionnelles. Cependant, nous butons ici sur deux obstacles immédiatement. Tout d'abord, quid des machines en tranches que nous venons de voir ? Un même jeu de chips permet d'aller de 4 à 32 bits par incrément de quatre bits (et même de deux bits pour des μP encore plus particuliers dont nous parlerons plus loin). Ensuite, pour une même longueur de mot nous observons des microprocesseurs assez disparates...

Un second critère — important sur le plan performances — est celui de la technologie utilisée : MOS canal-p, MOS canal-n, bipolaire, etc. A lui seul, il pourrait constituer une base défendable de classement.

Un troisième critère nous est fourni par la destination privilégiée des μP : contrôle de processus, calculatrices de poche programmables ou non, communications, etc. .

Malheureusement, cette destination est souvent très floue, d'autant plus qu'on a à faire à des μP récents et puissants. Si les machines à 4 bits étaient — parfois — assez spécialisées, nous avons vu que dès qu'on atteint 8 bits le champ d'application s'élargit considérablement, surtout en canal-n.

Plutôt que de baisser les bras (ou lever le crayon !) nous utiliserons une classification assez hybride dont le seul mérite sera donc l'existence...

1) Microprocesseurs à 4 bits

Nous trouverons dans cette case à la fois les pionniers, les boîtiers calculateurs, les «Harvard» et les puces bon marché.

2) Microprocesseurs à 8 bits

- Sous-classe MOS canal-p : cette fois, ce sont des chips d'utilisation plus large, comme nous l'avons vu, mais à qui leur vitesse moyenne interdit plusieurs secteurs d'applications. Notons toutefois que certaines machines récentes de cette catégorie approchent de la suivante.
- Sous-classe MOS canal-n : c'est la catégorie vraiment universelle, d'usage le plus général, dont la rapidité et les facultés d'adressage sont telles que les applications envisageables deviennent innombrables.

3) Microprocesseurs en tranches, microprogrammés

- Sous-classe MOS : nous venons d'en voir un exemple. Dans la pratique, les utilisateurs se sont jusqu'ici bornés à utiliser le microprogramme standard mettant un jeu d'instructions à leur disposition.
- Sous-classe bipolaire : comme nous l'avions glissé dans le paragraphe sur le bipolaire Schottky, une famille de chips LSI utilisant cette technologie vient d'être mise sur le marché par ... Intel. Visant un domaine d'applications très différent, la série 3000 (dont nous parlerons plus soin) a déjà permis entre autres de réaliser des microprocesseurs fonctionnant à plusieurs *millions* d'instructions par seconde !

4) Microprocesseurs non classés

Dans cette catégorie, que nous n'avons pas fait figurer dans le grand tableau récapitulatif double page, entrent les μP les plus divers. Mais avant de les évoquer (très rapidement), quelques mots de commentaires sur le tableau.

D'une part, nous avons tenu à y mentionner des dates et des prix, pour fixer un peu les idées du lecteur, mais nous tenons à souligner que ce sont des éléments très imprécis. Les dates, car l'appréciation du moment précis d'arrivée d'un chip sur le marché est très délicate : les chaînes se mettent en route lentement en luttant contre des difficultés de fabrications considérables.

On a souvent tendance à oublier à quel point cette micro-électronique se situe aux frontières du possible : de nombreux μP ont été annoncés, décrits dans le détail... et n'ont pas pu être livrés (le très intéressant et ambitieux AMI 7300 par exemple, et pourtant AMI est en flèche dans ce domaine, puisqu'avec Mostek tous deux ont permis l'essor des fameuses calculatrices de poche scientifiques de Hewlett-Packard, de la HP-35, première machine du genre, à la HP-65, premier micro-ordinateur de poche programmable à cartes magnétiques. Les prix, eux sont difficiles à définir, les fabricants ayant des tranches dégressives en quantité différentes, et les fluctuations dues au rodage des chaînes et à la concurrence étant très importantes.

Enfin, les données de départ sur lesquelles nous avons travaillé sont imprécises, même à partir des documentations techniques des μP, vu l'évolutivité des produits ; quant aux tableaux US que nous avons utilisés, ils contenaient tous de très nombreuses erreurs. Tout ceci pour dire que la nôtre ne sera pas sans défauts... Examinez le en détail cependant, nous pensons que vous y trouverez des informations intéressantes.

Mais revenons à notre catégorie «non classés» : vous vous en doutez, c'est un fourre-tout. On y trouvera les machines suivantes :

— des μP 8 bits plus ou moins classiques, récents et encore peu connus :
  - Rockwell PPS-8. Grand frère du PPS-4, P-MOS mais d'une rapidité fort convenable due à son architecture particulière riche en innovations «sophistiquées».
  - Electronic Arrays. N-MOS, avec opérations sur chaînes de 16 octets.
  - Fairchild F-8. N-MOS, d'intéressantes singularités : l'horloge est sur le chip CPU lui-même, mais surtout un CO existe sur chacun des chips d'interfaçage mémoire, ce qui facilite le traitement des interruptions.
  - AMI S6800. Il est identique au Motorola, 6800, constituant donc une «seconde source» du produit.
  - National CMP-8. Toujours 8 bits N-MOS.

— un μP unique dans sa catégorie, à mot de 12 bits : le Toshiba TLCS-12. Une CROM contient

Mot	Technologie	Ne figurent ni les µP livrables en 1975 ou après, ni les µP spéciaux	HARD									
			Nombre de chips	Nombre d'instructions	Temps en µs add. Reg. à Reg.	Arithm. DCB	Capacité d'adressage	Longueur mot		Registres		
		Marque et Type						Instruction	Donnée	Accum.	Index	Général
I 4 bits	MOS P	FAIRCHILD PPS 25		32 + 63 I/O	3	oui	6 656	12	4	1		
		INTEL 4004	1	45	8	oui	4 Ko	8,16	4	1	0	16
		INTEL 4040	1	60	8	oui	8 Ko	8,16	4	1	0	24
		ROCKWELL PPS 4	1	50	5	oui	4 Ko	8	4	1	0	1 + 3
II 8 bits	MOS P	INTEL 8008	1	48	20		16 Ko	8, 16, 24	8	1		6
		INTEL 8008-1	1	48	12,5		16 Ko	8, 16, 24	8	1		6
		MOSTEK 5065	1	51	10		32 Ko	8	8			
		ROCKWELL PPS 8	1	109		oui	16 Ko	8, 16, 24	8	1	1	2
	MOS N	INTEL 8080	1	78	2	oui	64 Ko	8, 16, 24	8	1		6
		MOTOROLA 6800	1	72	2	oui	64 Ko	8, 16, 24	8	2	1	
III Tranches	MOS P	NATIONAL GPC/IMP	3	43	6	poss.	4/64 Ko	4/16	4/16	2/4	0/2	0
	Bipolaire	INTEL Série 3000	3	var.	0,3 (16 b)	poss.	var.	2 n	18 +	1		11

		SOFT					Prix *approximatif* en $ unitaire par 100	Maturité (approx.)		Les informations prix et dates sont *très* fluctuantes : données seulement comme ordre de grandeur.
DMA	Interruptions	Assembl. résident.	Cross assembl.	Moniteur	Simulateur	Langage évolué		Premiers Échantillons	Premières Livraisons	Remarques Particulières : Malgré tous nos soins, les données sont si discordantes que le tableau contient certainement des inexactitudes.
	oui						60		72	Chip de calculateur de bureau 4 opérations, très évolué. Opérandes en chaîne de 25 caractères DCB en 62,5 μs.
	non	oui	oui	oui	oui		30	2 T 71	4 T 71	1er microprocesseur d'usage général fabriqué.
	oui	oui	oui	oui	oui		40	4 T 74	4 T 74	
	non	oui	oui		oui		45		73	Traitement de chaînes de caractères DCB très efficace. Famille de chips évolués très complète.
	oui oui	oui oui	oui oui	oui oui	oui oui	oui oui	45 60	4 T 71	1 T 72	
oui	oui	oui	oui				58	1 T 74	3 T 74	Le CO et l'accu sont en triple exemplaire, formant 3 CPU pour une même ALU sur un seul chip.
	oui	oui	oui				47	4 T 74	1 T 75	
oui	oui	oui	oui	oui	oui	oui	150	4 T 73	2 T 74	1er N-MOS + CPU statique
oui	oui	oui	oui		oui		150	2 T 74	4 T 74	
poss.	oui	oui	oui	oui		oui	100/250	73	74	Dates échelonnées (chips ou μP avec μ instructions standard).
oui	oui	Cross Microprogramming System					500	3 T 74	3 T 74	1er bipolaire en tranches. Microprogrammable par l'utilisateur.

128 micro-instructions standard de 29 bits. Six registres généraux, plus CO et PSW. Il y a 80 instructions, temps d'exécution de registre à à registre 3 μs, 8 lignes d'interruption. Prix par cent : 200 dollars, livrable.

— des machines en tranches nouvelles :
  • Monolithic Memories 6701, bipolaire, RALU de 4 bits,
  • Raytheon RP-16, idem, rapidité moyenne,
  • Transiston TMC/16001, idem, deux fois plus large que l'Intel 3000 et 2,5 fois plus lent,
  • Texas Instruments SBP0400, dont nous parlerons en 3-9 car il est bipolaire I^2L.

— deux μP en technologie C/MOS, l'Intersil 6100 et le RCA COSMAC, tous deux intéressants.

— enfin, trois μP 16 bits ! Nous en parlerons en 9. Et maintenant, un peu de pratique amusante suivie d'études plus en profondeur.

3 - 3. COMMENT L'ESPRIT DEPLACE LA MATIERE

Histoire de faire rapidement toucher du doigt la réalité concrète des microprocesseurs en action, nous allons faire nos premiers pas en psychokinésie appliquée, fidèles en cela à nos retours réguliers vers le réel.

A maintes reprises, glissant sur les applications non dénombrables des μP, nous avons vu le relent de *contrôle de processus* (ou de procédé, nous vous laissons le choix du vocabulaire) qui s'en dégage depuis le début. Et d'ailleurs, ce qui est finalement de même nature, il va bien falloir, outre les calculs, que nous commandions le mouvement de nos unités périphériques, aussi matérielles s'il en fût que les petites cuillères d'Uri Geller : une différence, avec le μP ce n'est pas du simple illusionisme travesti en PK, il va réellement transformer en action sur le monde extérieur les séquenes d'instructions de programme sorties tout droit de notre esprit.

Nous sommes gardien de zoo, et la haute direction, soucieuse d'économie, nous refuse l'investissement nécessaire, en compteurs industriels électromécaniques ou bipolaires coûteux, pour automatiser le tourniquet d'entrée : il nous faut compter un à un au passage les enfants de chaque groupe en visite. Mettons donc à profit ce que nous avons appris en logique et en électronique, et avec un facile montage (qui en plus résoudra nos problèmes de tenue de caisse) nous allons pouvoir nous en tirer aux moindres frais. Un chèque de quelques centaines de francs seulement à Heathkit nous procure le kit (19) de la calculatrice électronique de table IC 2108, avec une belle visu Sperry à décharge gazeuse. Les huit chiffres oranges à sept segments ont 12,7 mm de hauteur : l'institutrice qui accompagne le groupe pourra lire de loin le nombre d'entrées que nous y afficherons pour la commande du tourniquet.

On voit sur la figure que le microprocesseur (le boîtier calculateur TMS-0119 de Texas Instruments, dont nous avons déjà parlé) est relié d'une part au clavier, touches numériques et de fonction, et d'autre part à la visu : c'est vers elle que partent les «commandes» résultant du calcul, signaux que nous allons utiliser, justement, pour commander la libération ou le blocage du tourniquet. Un électro-aimant, à noyau plongeur servant de pêne, ne le libérera que si le courant est appliqué. Une ampoule électrique, un disque à deux fentes et deux photo-résistances, complètent l'électronique de notre «périphérique», permettant de repérer ses révolutions (le disque souple avec son ou ses trous montre — si l'on ose dire — le bout de l'oreille...).

Première manœuvre : nous frappons au clavier le signe *moins*, puis le nombre de personnes à laisser passer. Le chip μP illumine alors la visu au moyen de dix fils reliés aux caractères pouvant être affichés (8 + point décimal + signe moins) et de sept fils reliés chacun à tous les segments ayant le même emplacement dans les chiffres (segment A horizontal supérieur, B en haut et C en bas à droite, D en dessous, E en bas et F en haut à gauche, G au milieu). Un balayage régulier d'impulsions parcourt de droite à gauche les dix fils : un segment s'allume si la tension est appliquée à la fois sur son propre fil et le fil du digit.

Que se passe-t-il dès qu'une personne franchit le tourniquet ? Nous avons câblé les deux photo diodes en parallèle avec les touches 1 et +, d'un coup de soudure derrière le clavier : au premier demi-tour, la fente extérieure laisse passer un instant la lumière qui tombe sur la photo-résistance de droite. Comme nous l'avons vu en parlant des semiconducteurs et de l'action des photons, elle devient conductrice tant qu'elle est éclairée. Tout se passera alors comme si nous avions appuyé le même temps sur la touche 1. Au second demi-tour, ce sera la touche +, ce qui déclenchera une addition dans notre μP, et si nous avions affiché — 17 initialement, nous aurons — 16.

Vous l'avez deviné, c'est le segment G de gauche qui est la clé... du tourniquet. Son action se matérialise au travers d'un boîtier supplémentaire rajouté au système, appartenant à la famille C/MOS de RCA, la série 4000. Ces DIP présentent l'intérêt de la simplicité d'utilisation (une seule tension d'alimentation, de + 3 à + 15 volts) accompagnée des qualités du C/MOS, dont nous n'avions vu qu'une seule, la très faible consommation. Mais c'est également une autre qui nous intéresse ici : l'excellente immunité au bruit. Le C/MOS 4013A contient deux bascules D, vieilles connaissances donc (nous n'en utiliserons qu'une seule). Tant que l'impulsion d'horloge C et celle de

(19) Il s'agit d'un véritable kit, comprenant tous les circuits imprimés prêts à recevoir les composants, la visu, le clavier, l'alimentation, le coffret, etc. : la conception du matériel est faite. Attention : les «kits d'évaluation» de uP mis sur le marché par certains fabricants de composants comprennent, eux, simplement une poignée de chips accompagnés des feuilles techniques de caractéristiques. C'est au malheureux acheteur à se débrouiller seul pour savoir quoi en faire !

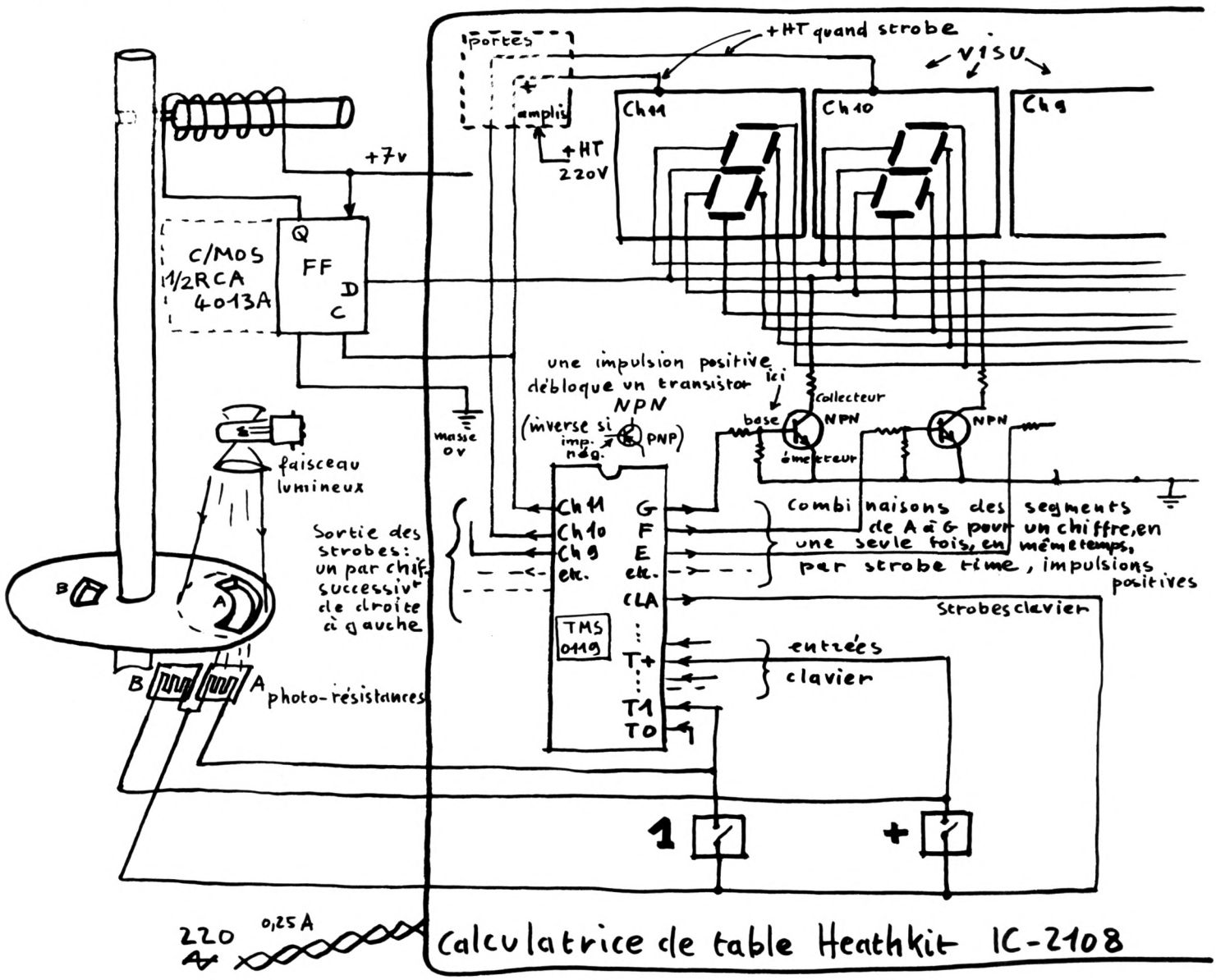
l'entrée D sont à 1 en même temps, Q est à 1, c'est-à-dire à la «haute tension» de + 7 volts : aucun courant ne passe donc dans l'enroulement. Dès que la dernière personne est passée, le compte affiché tombe à zéro avec disparition du signe moins : horloge à 1, D à zéro, la sortie du flip-flop de $Q_t = 1$ se change en $Q_{t+1} = 0$, c'est-à-dire qu'elle est directement en communication avec la masse par un transistor FET devenu conducteur. Le courant passe donc, et le solénoïde attire le pêne dans la gâche, bloquant le tourniquet par une victoire méritée de l'esprit sur la matière.

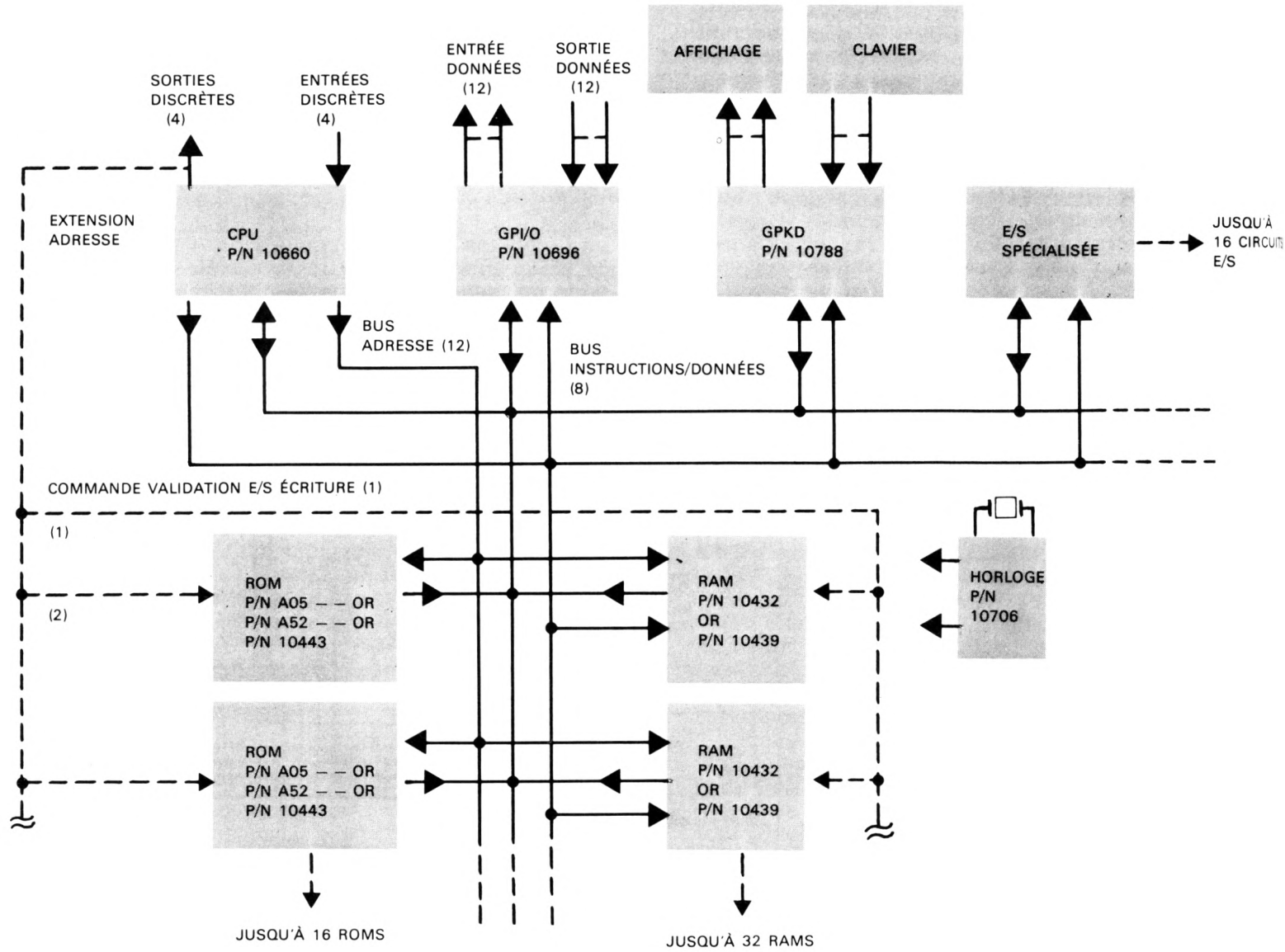
Certes, ici ce n'est pas nous qui avons pensé le programme inscrit en ROM, maître d'œuvre des opérations arithmétiques, rafraîchissement de la visu, envoi de l'impulsion d'exploration du clavier par CL, et autres. Mais le principe est là... et bientôt ce sera à notre tour d'agir.

3 - 4. LE PARALLEL PROCESSING SYSTEM 4

Sous ce nom ronflant se cache notre bon vieux PPS-4. C'est le premier des trois microprocesseurs que nous allons décrire un peu plus dans le détail, trois μP sélectionnés dans les principales classes qui sont d'intérêt pour nous :

— un 4 bits bien «typé» dont nous avons déjà annoncé les qualités : famille très complète de chips, performances acceptables et coût extrêmement bas, sans oublier une documentation agréablement détaillée (alors que celle du PPS-25 de Fairchild, déjà doté de l'architecture la plus obscure du marché, le fait paraître encore plus embrouillé qu'il ne l'est !),





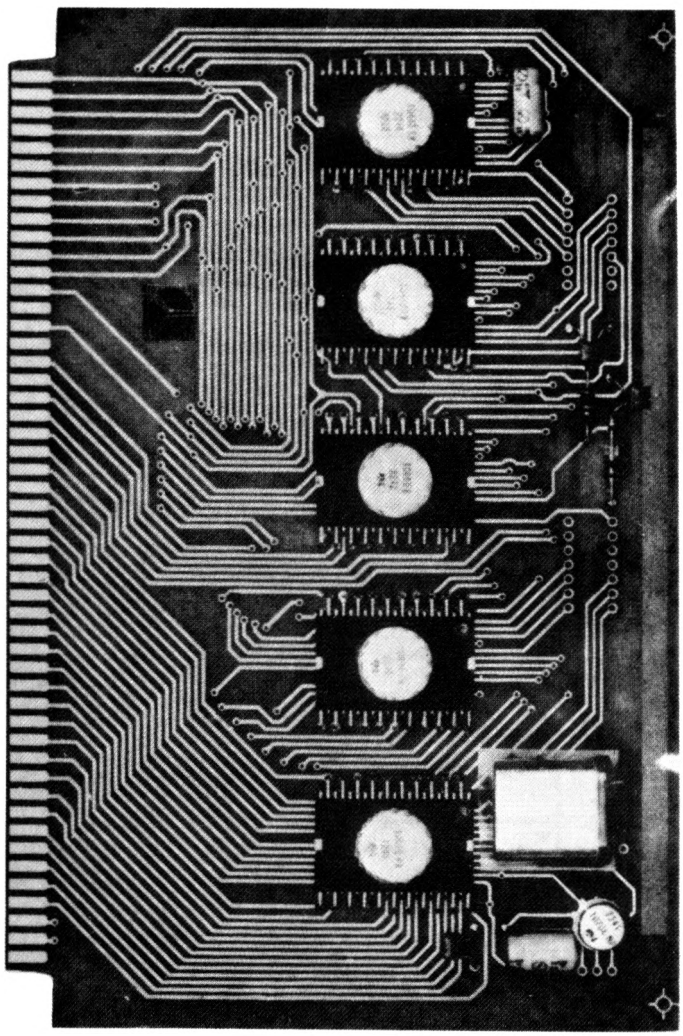
— un 8 bits de première génération, et
— un 8 bits de seconde génération.

Le PPS-4 est donc mi-Harvard mi-Princeton, comme nous l'avons vu, et sans plus commenter architecture et adressage que nous référons au premier croquis indiquant l'organisation générale d'un système complet. Le chip CPU peut être relié directement à une trentaine d'autres chips de la famille, ce qui est excellent. Notons d'ailleurs qu'il est conditionné dans un *flat-pack* à 42 sorties, comme tous les autres membres de sa famille (sauf l'horloge en boîtier circulaire standard T0100) :

— chip général d'entrée/sortie, avec trois groupes de quatre bits en entrée, et trois autres en sortie. Il fonctionne par réception d'une instruction de 8 bits venant de la ROM sur ordre du CPU. Le code opération de 4 bits indique le groupe concerné et le sens de l'échange avec l'accumulateur (les 4 autres bits constituent l'adresse du chip E/S : on peut donc en connecter 16),
— chip de gestion clavier/affichage, doté d'un décodage d'adresse et de commande, d'un compteur de balayage produisant les signaux nécessaires à l'exploration d'un clavier à 64 touches et au rafraîchissement d'une visualisation, comme dans l'IC 2108, avec l'aide d'une mémoire tampon d'un caractère de 9 bits (clavier) et de deux registres de 4 x 16 bits (visu),
— chip contrôleur d'imprimante Seiko à tambour,
— chip universel de transmission/réception synchrone/asynchrone USART pour liaison modem, CRT, etc.,
— chips buffers sur le bus,
— horloge à 199 kHz (division par 18 de la fréquence d'un quartz de TV couleur à 3,579545 MHz),

— chip RAM de 256 mots de 4 bits pour les variables, «registres» externes,
— chips ROM (attention : seulement programmables par masque à l'usine) de 1K et 2K octets pour le programme figé,
— un chip combiné ROM/RAM (704 octets ROM, 76 caractères de 4 bits RAM) permettant de constituer simplement un petit système complet.

On le voit, la liste est fournie, mais on sent bien que tout est orienté vers les ROM masquables en usine pour les grandes séries de micro-systèmes identiques spécialisés. La famille ne comprend pas de PROM, on doit aller les chercher à l'extérieur. Les PROM (Programmable Read Only Memory, mémoire programmable à lecture seulement) sont un élément indispensable pour remplacer les mémoires ROM lors de la mise au point des prototypes, ou pour de petites séries de machines (jusqu'à une bonne centaine ou même plus) ou lorsque des modifications de programme sont à prévoir. Inventés par... Intel, on peut y inscrire l'information binaire en quelques minutes par des impulsions électriques, et les effacer aux rayons ultra-violets.



Nous n'indiquerons pas le détail du CPU, bornons-nous à dire qu'il comprend un accumulateur de 4 bits et une pile de registres (extensible en RAM) pour les adresses retour de sous-programmes.

Le cliché représente une carte d'évaluation fournie par Rockwell, destinée à la mise au point des prototypes des clients, et comportant CPU, interfaces et un peu de RAM, le tout à relier à une mémoire PROM extérieure et aux périphériques. De haut en bas sur la carte on distingue :
— deux chips RAM de 256 x 4 bits,
— le chip CPU au centre,
— deux chips généraux d'entrée/sortie (GPI/0),
— tout en bas à droite l'horloge, avec au-dessus d'elle le boîtier métallique du quartz.

La carte comprend également le circuit d'initialisation à la mise sous tension : elle est donc bien complète, il ne manque que la ROM. On voit les lignes d'entrée/sortie émanant des GP1/0 pour aboutir aux plots de connexion à gauche (il y a une deuxième rangée de plots sur l'autre face, et les trous métallisés de passage sont nettement visibles) : 32 entrées et 28 sorties pour les 2 GPI/O et le CPU, occupant les 2/3 du connecteur.

Pour en finir avec les mystères du PPS-4, un mot sur la fameuse instruction dont nous avons parlé à la fin de «les pionniers, 4 à 4»... Il s'agit de EXD, l'une des principales des 50 instructions exécutables. Elle sert à référencer la mémoire RAM, l'adresse de départ d'une chaîne de caractères de 16 chiffres DCB de 4 bits ayant été chargée dans le registre B. L'addition sur un caractère venant d'être faite dans l'accumulateur, elle va échanger ce dernier avec le caractère DCB venant d'être additionné, puis effectue un OUX immédiat entre 3 de ses propres bits et les 3 bits de rang 5, 6 et 7 du registre (ce qui permet de repasser à l'adresse de la 1re chaîne opérande, décalée de n fois 16 chiffres, l'instruction de chargement qui précédait ayant elle-même par OUX incorporé fait passer de la 1re à la 2e, tout ceci par un véritable jeu de bascule), décrémente B de un (ce qui assure la progression le long de la chaîne) et inspecte la valeur des 4 bits de droite de B : s'ils vont de 1110 à 000, on passe à l'instruction suivante (un branchement qui renvoie au sommet de la boucle d'addition), mais s'ils sont devenus 1111 (franchissement de zéro), on est arrivé au bout du «registre RAM», et l'instruction suivante sera sautée, ce qui permet de sortir de la boucle.

Pas mal, hein, pour une si petite puce ! On comprend pourquoi 440 μs suffisent dès lors pour une addition décimale de 16 chiffres...

3 - 5. LE 8008, PIONNIER SOLIDE AU POSTE

Nous avons déjà beaucoup parlé de ce huit bits, de sa philosophie de conception et des conséquences qu'elle a entraînées pour son bus et l'électronique externe en particulier, aussi passerons-nous directement à une description plus formelle de l'engin.

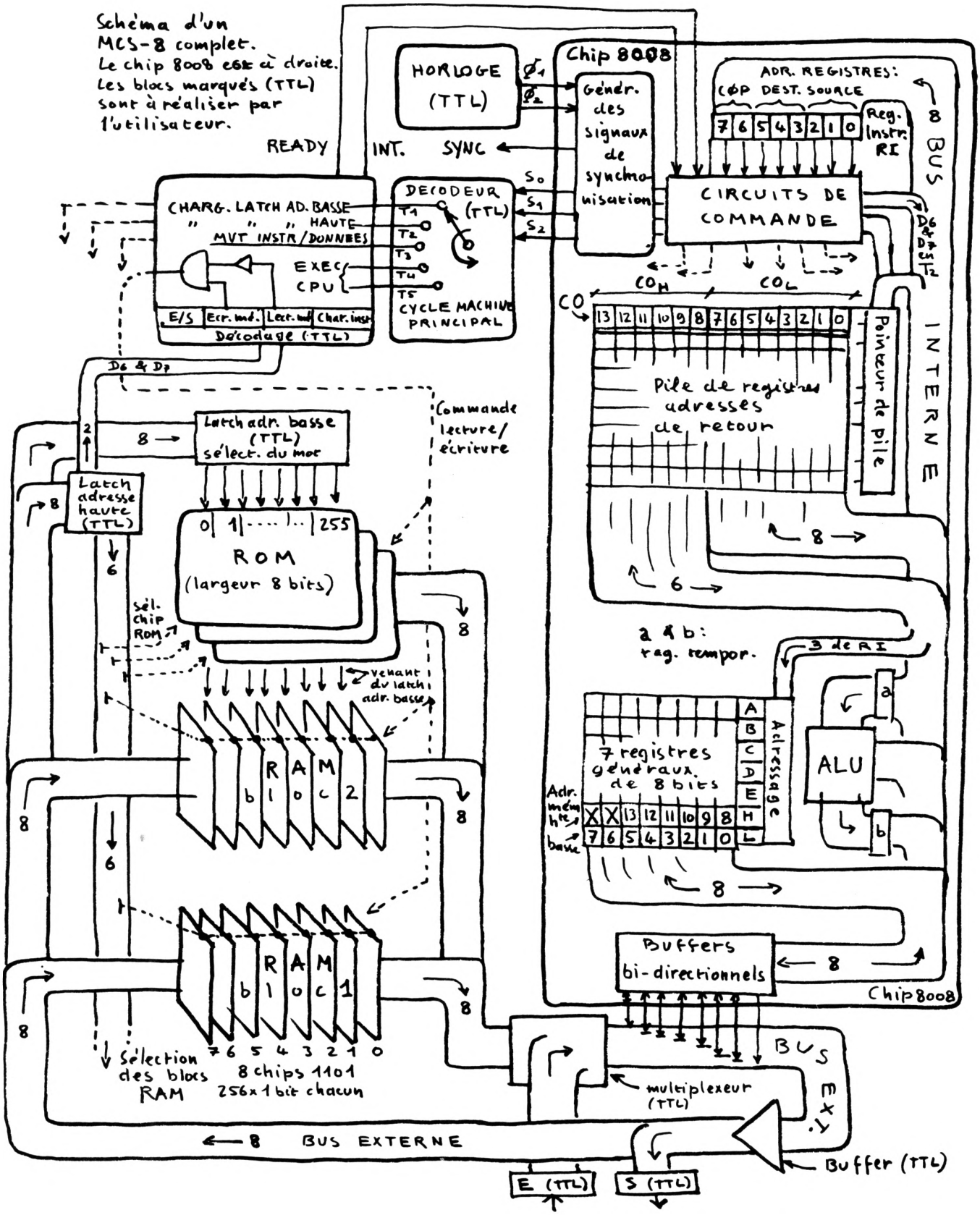


Schéma d'un MCS-8 complet.
Le chip 8008 est à droite.
Les blocs marqués (TTL) sont à réaliser par l'utilisateur.
HORLOGE (TTL)
Chip 8008
Génér. des signaux de synchronisation
READY
INT.
SYNC
ADR. REGISTRES:
C∅P DEST. SOURCE
Reg. Instr. RI
8 BUS INTERNE
CIRCUITS DE COMMANDE
CHARG. LATCH AD. BASSE
" " " HAUTE
MVT INSTR/DONNEES
EXEC CPU
E/S
Ecr. mé.
Lect. mé
Char. inst
Décodage (TTL)
DECODEUR (TTL)
T1
T2
T3
T4
T5
CYCLE MACHINE PRINCIPAL
S0
S1
S2
CO
COH
COL
Pile de registres adresses de retour
Pointeur de pile
Commande lecture/écriture
Latch adr. basse (TTL) sélect. du mot
Latch adresse haute (TTL)
ROM (largeur 8 bits)
sél. chip ROM
a & b: rag. tempor.
3 de RI
7 registres généraux de 8 bits
Adressage
ALU
Adr. mémoire
Buffers bi-directionnels
RAM bloc 2
RAM bloc 1
Sélection des blocs RAM
8 chips 1101 256x1 bit chacun
multiplexeur (TTL)
BUS EXTERNE
Buffer (TTL)
E (TTL)
S (TTL)

Sur le chip lui-même nous trouvons les éléments suivants :
— un accumulateur de huit bits (adresse A),
— sept registres généraux de huit bits (adresses B, C, D, E, H, L),
— quatre bits de conditions (flip-flops Z, C, P),
— une pile de huit registres adresse de 14 bits chacun, accompagnée de son pointeur,
— un registre instruction de huit bits,
— les circuits de commande qu'il déclenche,
— les générateurs de signaux de synchronisation qui leur sont associés,
— une unité arithmétique et logique flanquée de deux registres temporaires,
— un bus interne large de huit bits reliant tout ce beau monde ensemble, et débouchant sur un buffer (attention : au sens électronique, à la fois mémoire tampon et amplificateur de 1 ou n bits) bi-directionnel de huit bits où vient se raccorder le bus externe (ce buffer débouche donc sur huit broches du boîtier).

Deux broches amènent les + 5V et — 9V de l'alimentation, et deux autres les signaux Φ 1 et Φ 2 de l'horloge réalisée en TTL.

La plupart des instructions ont un octet de long, sauf celles à opérande immédiat (2 octets) et celles de saut (3 octets) : les codes opérations 10 et 00 servent essentiellement aux 28 instructions faisant intervenir l'accumulateur, les registres généraux et la mémoire (3 sont logiques et 4 de décalages), 7 instructions concernent les registres généraux (baptisés un peu abusivement «registres d'index» entre eux et avec la mémoire, il y a 2 instructions d'entrée/sortie permettant d'adresser 8 registres externes d'entrée de 8 bits et 24 autres de sortie de 8 bits également, l'instruction de halte et pas moins de 10 instructions de branchement et de call.

Le compteur ordinal (ou registre adresse instruction suivante, ou registre programme) courant CO est celui du haut de la pile, laquelle sert à entrer et sortir facilement dans les sous-programmes imbriqués (en contrôle de procédé, les programmes contiennent un fourmillement de petits sous-programmes qui s'appellent les uns les autres jusqu'à une bonne profondeur). L'instruction CAL d'appel de sous-programme enfonce la pile d'un cran en logeant en tête l'adresse de branchement qui devient le nouveau CO, l'instruction de retour RET agit en sens inverse et remet l'ancien CO en place. Tout ceci n'est pas si mal, nous avons affaire à un véritable ordinateur. Attention cependant à ne pas dépasser la profondeur de la pile (il n'y a pas de détection de débordement du pointeur) ce qui conduirait à des erreurs d'apparence étrange.

3 - 5 - 1. Bizarreries

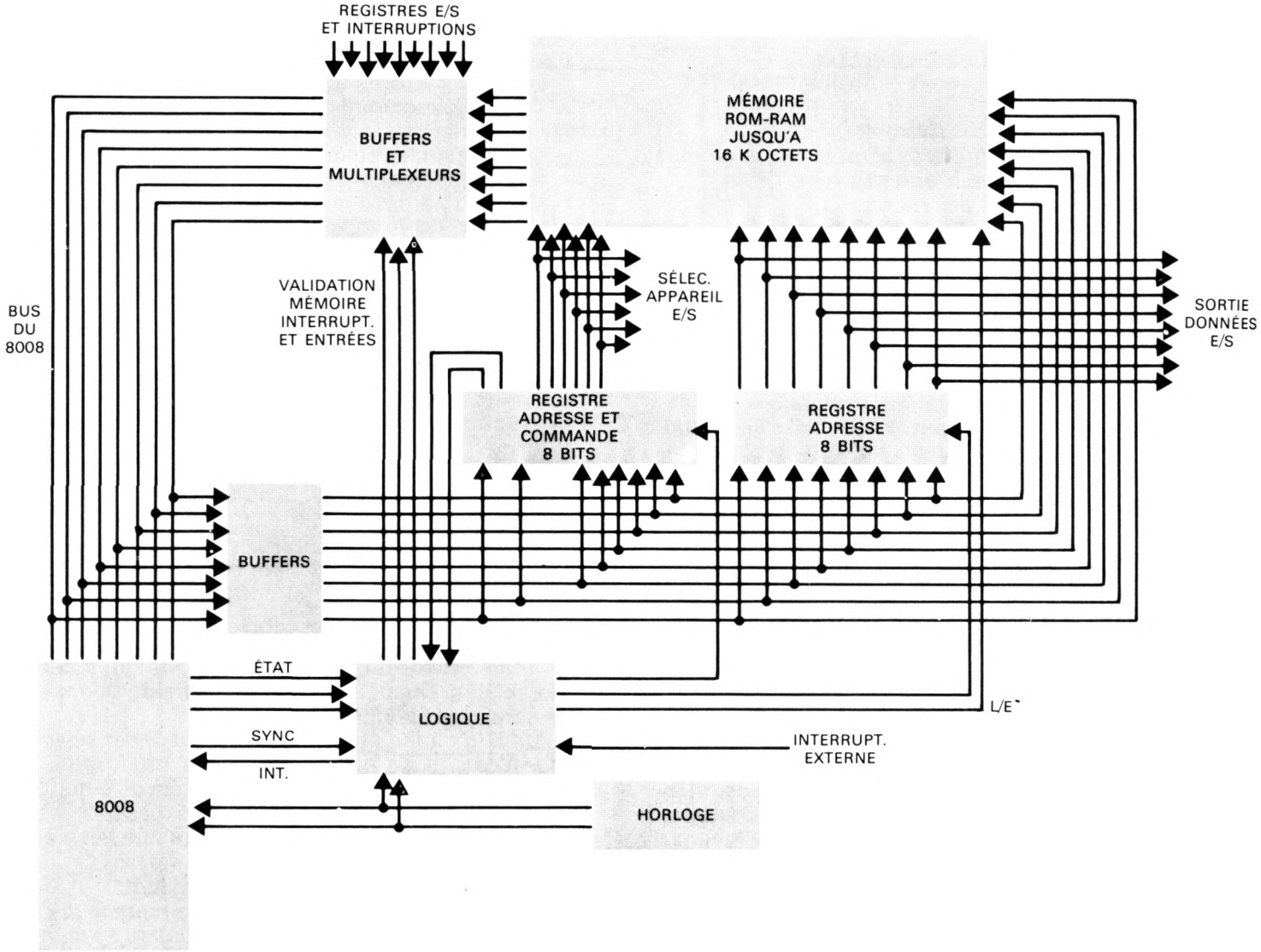
Les choses se gâtent un peu quand on découvre certains détails : comment adresser une donnée en mémoire (ou comme le dit savoureusement la brochure «un registre mémoire», ce qui fleure bon ce monde exotique où notre bonne mémoire RAM est réduite au rôle d'un minuscule accessoire stockant quelques opérandes... et pas de programme) ? En effet, avec une instruction de 8 bits et une taille mémoire pouvant atteindre 16K octets il y a comme un problème...

On ne marche en fait qu'en adressage indirect, à l'aide des registres H et L où l'on stocke au préalable l'adresse mémoire : il faut donc penser «adresse» avant «opération», ce qui permute les habitudes. L (*low,* bas) contiendra les 8 bits de droite de l'adresse, et H les 6 bits de plus fort poids, les deux bits de gauche de H n'intervenant pas pour l'adressage indirect. Si l'on écrit alors l'instruction ACM (code machine 10 001111) on additionne dans l'accu le contenu de l'octet en Mémoire désiré. Fort heureusement, il existe des instructions d'incrémentation et de décrémentation des registres «d'index» permettant de progresser rapidement les adresses pour traiter les chaînes de caractères. Enfin, ce n'est pas le Pérou, et si l'on n'étudie pas soigneusement les positions des paires d'opérandes les uns par rapport aux autres, en particulier pour rester dans des pages communes de 250 octets, ce qui évite de recharger H, on tombe dans une épicerie sans nom ! Il vaut mieux systématiser les opérations en utilisant surtout (A) + (B) → (B), ce qui facilite les choses. Quant aux déplacements de données, ils sont assez pénibles.

Mais là où les choses deviennent assez obscures, c'est en ce qui concerne les interruptions. Une première approche laisse perplexe. Puis on réalise progressivement ce qui se passe : lorsqu'une interruption arrive, tout ce qu'elle fait c'est seulement d'empêcher l'incrémentation de CO après la recherche instruction suivante. Il faut alors qu'une logique TTL externe entonne dans le bus une instruction appelant le sous-programme correspondant au type d'interruption qui s'est produit, le CO courant étant alors enfoncé d'un cran dans la pile comme vu. A partir de ce moment, des difficultés sérieuses surgissent : aucun stockage automatique des registres généraux en mémoire ne s'effectue, et il faut pourtant dégager assez rapidement de la place sans écraser d'information dans les registres, ni encourir la pénalité d'en réserver systématiquement un certain nombre de libres, inutilisés, pour enregistrer les données résultant de l'interruption. Une bonne quantité de TTL externe supplémentaire bien pensé permet de résoudre le problème, mais la plupart des utilisateurs ne se servent en définitive de l'interruption que pour démarrer le μP au moment de la mise sous tension.

3 - 5 - 2. Un bon cheval quand même

Tous ces reproches ne sont pas fondamentaux et ce que peut accomplir ce si petit engin à 8 bits reste tout bonnement extraordinaire.



Nous ne voulons pas commenter trop longuement les deux croquis complémentaires indiquant l'architecture interne du 8008 et externe d'un MCS-8 : le lecteur y trouvera une foule de précisions concernant les méthodes de sélection, d'adressage, etc., et nous lui conseillons de les analyser dans le détail, surtout après avoir lu les explications qui viennent ci-après.

Le cycle instruction de base, ou cycle machine du 8008, comprend normalement cinq états successifs, de T_1 à T_5, durant chacun une période du signal SYNC qui synchronise tout l'extérieur du 8008 et est lui-même déduit de Φ_1 et Φ_2 qui pilotent l'ensemble : ceux qui connaissent le détail du fonctionnement de leur ordinateur (ou ont simplement lu *Un Fil d'Ariane* : tome I, et n'ont qu'à rechercher le mot *cycle* dans l'index de 800 termes informatiques) remarqueront qu'il y a beaucoup d'états successifs. Ceci est dû bien sûr aux limitations que nous avons vues à propos du bus dont l'étroitesse nécessite de fractionner certaines opérations.

En T_1, la partie inférieure de l'adresse contenue dans le CO est envoyée sur le bus (inutile de préciser lequel, puisqu'il n'y en a qu'un seul, multiplexé), en T_2 c'est la partie gauche : la mémoire a alors reçu une adresse complète, et l'octet désigné va pouvoir être transféré, en entrée si c'est l'instruction suivante à exécuter (ROM en général), en entrée ou en sortie si c'est une donnée. Ce transfert s'accomplit pendant T_3, et pendant T_4 et T_5 le reste de l'instruction s'effectue de façon interne sur la puce (ALU, registres, etc.).

La question évidente est : comment la partie extérieure du système va-t-elle être informée de ce qui devra se passer pendant T_3 ? Les malins auront bien remarqué que la partie gauche du CO ne fait que 6 bits, et que l'on voit sur le grand croquis deux bits, D_6 et D_7, sortir des circuits de commande et s'enfiler subrepticement sur le bus en T_2.

Ils vont être expédiés vers le monde extérieur à l'écoute du CPU, et enregistrés dans des *bascules à verrouillage* (bistable latch, ou *latch* tout court) qui sont ni plus ni moins des mémoires tampons.

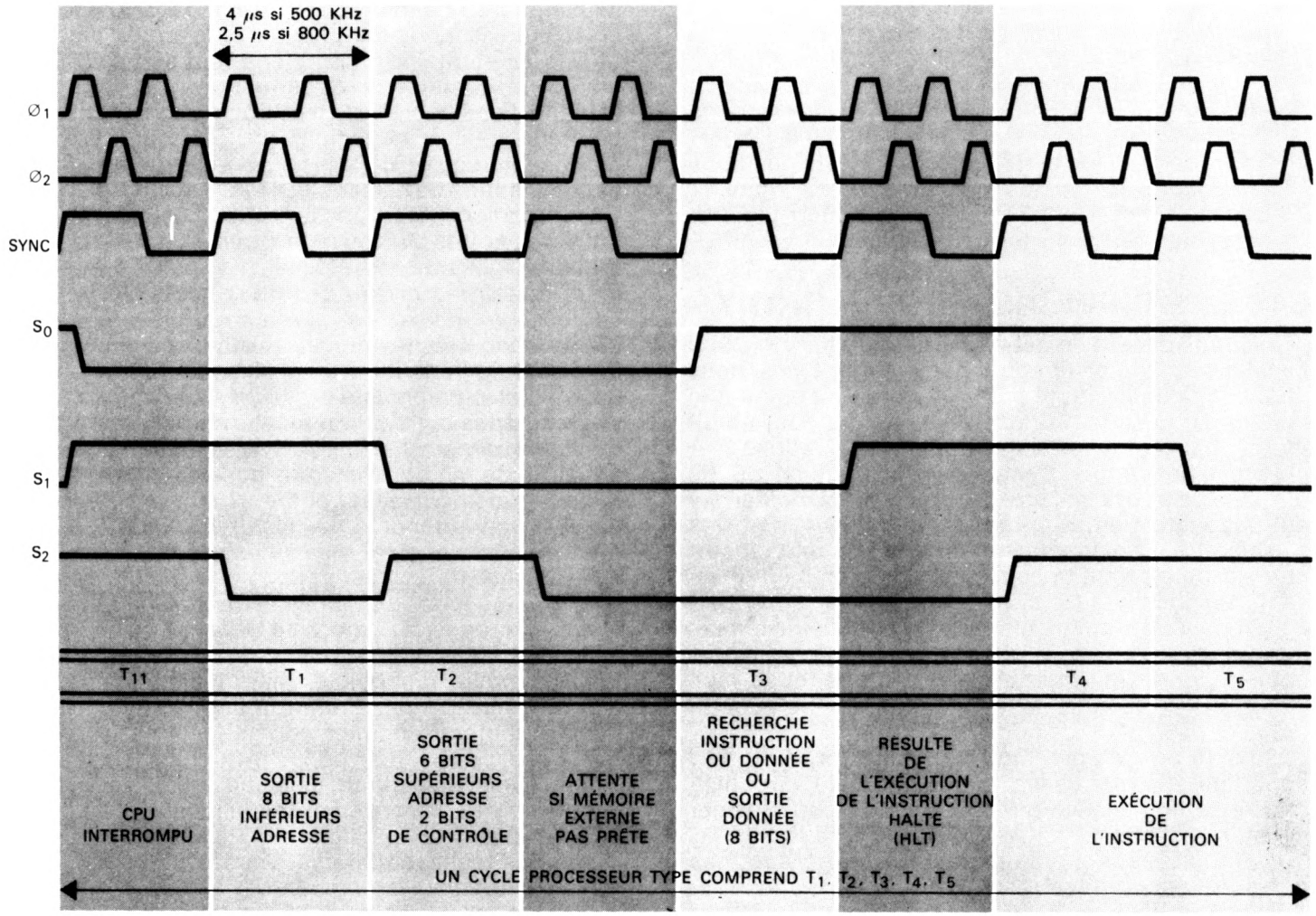
De là, proprement reliés à des portes logiques, ils contribueront à engendrer les signaux activant en particulier les lignes qui déclenchent les opérations de lecture/écriture mémoire, et celles d'entrée/sortie périphériques, soit 4 situations possibles en T_3. La présence des latches un peu partout est essentielle, vu la présence fugitive des signaux sur le bus multiplexé.

Toutes ces phases s'effectuent en série, le nombre d'états successifs dépendant de la complexité de l'instruction. Il y en a 5 comme nous venons de le voir pour une instruction de registre à registre, 8 si la mémoire intervient (deux d'adresse et un de transfert en plus) et 11 pour l'instruction la plus compliquée, CAL.

On remarque sur le schéma du cycle instruction qu'il existe trois états dont nous n'avons encore pas parlé :

— état WAIT : c'est l'attente, qui se déclenche après l'envoi d'adresse, jusqu'à ce que la mémoire ait retourné un signal indiquant qu'elle est prête : l'état WAIT peut durer beaucoup plus qu'une période. Il permet donc d'utiliser des mémoires externes de rapidité plus faible si on le désire. Avec des chips mémoires de vitesse suffisante, l'état WAIT ne se manifeste pas ;
— état STOPPED : sur exécution de l'instruction HLT, le CPU (le processeur) stoppe et ne redémarrera que sur activation de l'interruption ;
— état interrompu, en attente d'exécution d'une instruction de redémarrage RST (restart) qui effectue un call sur l'une des huit adresses spécifiées en page zéro (les 256 premiers octets en mémoire).

Les trois sorties, S_0, S_1 et S_2, indiquent sous forme d'un nombre binaire de trois bits l'état en cours parmi les huit possibles. Une logique TTL extérieure est donc nécessaire pour décoder



Les 8 états possibles (5 courants) du 8008 et le cycle instruction de base

l'état, le combiner aux bits D_6 et D_7, et de là commander les opérations extérieures.

Le 8008 accepte une fréquence d'horloge de 500 kilohertz, ce qui correspond à une durée d'état de 4 microsecondes (deux périodes d'horloge, donnant une période de SYNC). Une instruction de registre à registre dure donc 20 μs, une instruction référençant la mémoire 32 μs, et la plus longue de toutes 44μs. En pratique, la majorité des instructions d'un programme sont du premier type, le reste du second.

La dispersion des caractéristiques de fabrication des circuits intégrés permet, en triant les puces par tests, de sélectionner des circuits ayant de meilleures caractéristiques que la moyenne : c'est ainsi qu'est obtenu le 8008-1. Il accepte une fréquence d'horloge de 800 kHz, donc 2,5 μs de durée d'état, et des instructions courantes de 12,5 μs. Le gain est appréciable, et ouvre déjà un plus large champ d'action à la puce. Certains concepteurs de systèmes, même sans avoir besoin de cet avantage, préfèrent de toutes façons payer un peu plus cher pour utiliser le 8008-1 dont ils préfèrent l'horloge à deux phases symétriques (ce qui n'est pas le cas du 8008, on l'aura remarqué sur le schéma).

Tout bien pesé, on ne sait trop ce qu'on doit admirer le plus dans cette machine : l'ingéniosité des designers en face des contraintes de fabrication, son prix, les performances du μP... ou l'habileté déployée par les concepteurs de toute la population de DIPs TTL *devant* entourer le CPU, aussi bien que celle des programmeurs !

3 - 6. LES DIFFICULTES D'UN CHOIX

Avant de décrire en détail notre troisième μP (8 bits canal-n seconde génération, nous le rappelons) nous allons faire un bref retour en arrière qui va éclairer le lecteur sur la suite. Comment se décider pour un μP plutôt qu'un autre, en face de tant de chips variés ?

Lorsqu'au dernier trimestre 1974 l'équipe du Fil d'Ariane se trouva en face du choix pour Nanostar, il y eut des états d'âmes cornéliens. Le PPS-4 était très valable et bien disponible, d'un remarquable rapport performances/coût, et accompagné de chips intéressants (scanning du clavier, etc.) ; mais d'une part c'était un 4 bits, moins bien adapté à certains domaines, et d'autre part sa vitesse posait un problème pour la connexion ultérieure de périphériques rapides. Nous décidâmes de le garder en réserve pour un projet dérivé.

Côté 16 bits, c'était l'inverse. L'IMP-16C était trop important et cher, sans aucune famille de chips prévue (le nouveau chip FACE vient un peu changer cet état de choses).

Quant à la microprogrammation... on ne peut pas dire que c'est du gâteau !

Restaient donc les 8 bits, aucun d'entre eux n'ayant cependant de famille logique de chips aussi sérieuse que celles des 4 bits, avec leurs véritables contrôleurs de périphériques, dues aux investissements rendus possibles par l'énormité des séries dans le secteur calculatrices de poche. Le 8008-1 était solidement connu, d'un prix très abordable, et muni d'un excellent support software : assembleur fonctionnant avec le μP lui-même, cross-assembleur sur gros ordinateur et réseaux de temps-partagé, idem pour un langage évolué, le PL/M, sous ensemble de PL/I ! Par contre, les contraintes électroniques sont nombreuses, et un système réaliste demande une bonne vingtaine de boîtiers DIP en MSI TTL pour l'interfaçage du μP avec la mémoire et les entrées/sorties... et le travail de conception et de dessin de circuits associé. De plus, les 8 bits deuxième génération arrivaient, alors...

A ce point, la décision commença à devenir difficile. Le 8080 allait-il voir le Motorola 6800, nouveau venu, lui ravir la palme ? Le 6800 était déjà présenté partout comme muni d'une famille de chips compatibles TTL, très bien pensée (le PIA, Peripheral Interface Adaptor, en particulier) permettant de constituer un système complet avec 6 DIPs seulement, et une seule tension d'alimentation, + 5 V. Le CPU possède un registre d'index (ou plutôt d'adresse de base) de 16 bits, ses temps d'instructions vont de 2 à 12 μs, ce qui est respectable, il a des instructions *décimales*, et dans l'ensemble sa structure déroute moins les habitués des gros ordinateurs, ce qui n'est pas étonnant lorsqu'on découvre que le jeu d'instructions est très inspiré de celui du PDP-11.

Un examen plus poussé laisse cependant apparaître des points plus ou moins gênants : le CPU, sans registres généraux, présente un goulet d'étranglement certain au niveau des deux accumulateurs, et l'adressage indexé ne permet d'utiliser dans les instructions qu'une adresse relative couvrant $\pm$ 128 octets. Quant à la famille de chips, la PROM électriquement programmable (la seule praticable pour les petites séries de machines comme nous le verrons) était prévue pour 1976. Les systèmes à six boîtiers, après étude, ne se révélèrent possibles qu'avec des tailles de mémoire assez petites. Un assembleur avec un système autonome de mise au point très limité était disponible, ainsi qu'un beau cross-assembleur sur réseaux de temps partagé. La documentation, elle, commençait tout juste à arriver, et le centre de support européen se trouvait à Genève.

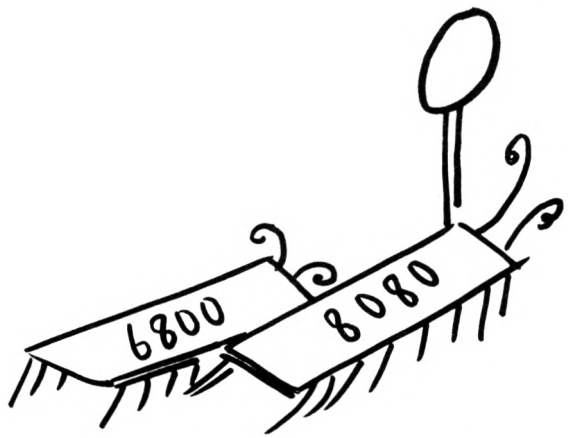
En face, le 8080, environ 30 % plus cher. Peu de famille encore disponible, mais les PROM (encore une première d'Intel) étaient là depuis longtemps. Il fallait compter dix boîtiers pour le système, mais avec de larges possibilités de connexions mémoires et autres. Quant au support, déjà une bonne documentation, un centre européen à Rungis, tout le soft écrit pour le 8008 (y compris donc le PL/M) et un système d'aide à la réalisation de software, l'Intellec 80, utilisant bien sûr un μP comme CPU.

Là où le 6800 présentait un avantage important par rapport au 8080, c'était pour la gestion des interrup-

tions : il ne demandait que 22 microsecondes pour l'ensemble des opérations (y compris la sauvegarde des registres) contre 52 microsecondes pour le 8080. Pour des applications industrielles, ce point est essentiel, mais pour nctre utilisation, il était relativement secondaire.

Genèse d'un fils spirituel

Enfin, la décision horizon 1975 fut prise, à l'unanimité : il faut bien voir qu'elle engageait assez sérieusement l'avenir, l'investissement en temps pour bien se familiariser avec un μP, sa famille logique de chips et tout le reste de l'environnement, étant loin d'être négligeable, et rendant



D'une courte tête ...

douloureux un changement de cheval au milieu du gué. Motorola et Intel donnaient identiquement les signes extérieurs indiquant qu'ils étaient tous deux fermement décidés à soutenir leurs microprocesseurs, ceci étant l'élément de jugement fondamental.

A égalité sur ce point, c'est donc l'avance technique d'Intel, leur maîtrise du procédé de fabrication, qui emportèrent la décision.

Six mois plus tard, Intel annonçait en juin 1975 que le prix à l'unité du 8080 passait de 2070 F (HT) ... à 950 F ! Simultanément d'ailleurs, le chip d'interfaçage si pratique, bonne à tout faire de la famille (malheureusement trop onéreux, et que nous avions passé longtemps à éliminer par des boîtiers TTL bon marché) baissait de plus de 40 %, et une famille respectable d'autres chips intéressants apparaissait. Les dessins de circuits furent jetés au panier, et quelques semaines plus tard Alcyane, premier membre de la famille Nanostar, seul fœtus à avoir subi un lifting avant sa naissance, présentait un aspect moins tourmenté à ses pères émerveillés...

3 - 7. NOTRE MILLE-PATTES DE BATAILLE : LE 8080

Le lecteur nous reprochera sans doute d'osciller au gré de la plume entre des images peu compatibles des μP : allons-nous oser dire maintenant que le 8080 est un très bon cheval ? Eh bien oui, pourquoi pas, Alcyane ne protestera sûrement pas !

C'est à la fois comme dérivé et différent du 8008 que se présente le 8080 (prononcez quatre vingts, quatre vingts, c'est l'habitude, et on peut le fredonner sur l'air du refrain des 80 chasseurs, qui n'avaient pas peur, dans le grand lit de la marquise, quatre vingts (do) quatre vingts (sol) quatre vingts (mi) quatre vingts (do)...).

La technologie MOS canal-n, une puce légèrement plus grande (4,2 x 4,8 mm), et surtout un conditionnement dans un DIP céramique à 40 broches, ont permis une progression considérable par rapport à la première génération. Quarante n'est-il pas pour les orientaux le premier nombre quasi infini, comme 36 pour nous, ainsi qu'en témoigne le palais des 40 colonnes à Esfahan ? En les comptant bien d'ailleurs on n'en trouve que 20, mais il y a aussi les 20 reflets dans la pièce d'eau...

3 - 7 - 1. Dix fois plus rapide

Dès que la chaîne de fabrication MOS canal-n à grille silicium fut en mesure de produire un volume élevé, vers mi-72, la décision fut prise chez Intel de donner un successeur au 8008. Le cahier des charges était net. Le 8080 devait :

— être 10 fois plus rapide,
— faire disparaître les limitations du 8008 (difficultés d'interfaçage, manque d'interruptions multiples),
— conserver les qualités du 8008, ainsi que ses instructions, mais en ajouter d'autres plus puissantes,
— être économique, c'est-à-dire utiliser un chip assez petit et un boîtier standard.

En ce qui concerne le premier but, la différence de mobilité des électrons par rapport aux trous n'était pas suffisante : 2,4 à 1 seulement. La tension de seuil plus faible du canal n permit de gagner au-delà, et l'amélioration des interconnexions, due à la grille silicium et au canal n que nous avons déjà explicités, joua également. Enfin, le désencombrement et l'élargissement du bus firent le reste. Ce dernier avantage tient également à la nouvelle technologie : alors que les 12 buffers du 8008 dissipaient au total 250 mW, les 30 buffers du 8080 n'en dissipent que 150, sont individuellement 6 fois plus rapides et quatre fois plus puissants, d'où des possibilités de connexions extérieures accrues. D'autre part, comme le bus d'adresses a toute la largeur voulue et est

séparé de celui des données, non seulement il peut communiquer l'adresse à l'extérieur pendant un seul temps T, mais il peut y rester pendant le temps T suivant, puisque les données passeront par leur propre bus : on élimine la nécessité des latches externes, les buffers du chip sont eux-mêmes les latches. Notons au passage, comme dernière information technique, que le boîtier choisi ne permet de dissiper que 750 mW : il ne faut pas faire des folies en concevant un μP, ce qui est bienvenu en ces temps de restrictions d'énergie !

Le dessin des masques du μP consomma 18 mois-hommes, et le soin pris permit de réaliser toutes les fonctions désirées avec seulement 5000 transistors matérialisant 1100 portes environ, la puce finale étant de dimensions inférieures à beaucoup des chips uniques de calculateurs de poche actuels.

Trente nouvelles instructions permettent de gagner plus de temps que ne l'indique une comparaison brutale des temps, et surtout de faciliter la programmation tout en réduisant de 10 à 30 % la longueur des programmes en moyenne. On peut dire qu'en pratique ces derniers se déroulent 10 à 20 fois plus rapidement.

3 - 7 - 2. Une nouvelle architecture

Les mémoires internes un peu disparates du 8008 sont regroupées ici en une structure plus nette : un jeu de registres larges de 16 bits et adressables individuellement en entier, ou par moitiés. Le CO occupe l'un d'eux, et cette fois la pile est en mémoire externe, sans limitation pratique de profondeur par conséquent. Elle est gérée par un pointeur de 16 bits compris dans nos 6 registres. Enfin, l'accu de 8 bits est flanqué d'une nouvelle ALU permettant beaucoup plus facilement les opérations décimales. Le registre instruction reste séparé, au voisinage des circuits de commande.

Ce groupage des registres, ainsi que le souligne Stanley Mazor, responsable de l'architecture, a entraîné immédiatement des bénéfices annexes. Le CO étant dans le jeu de registres, un incrémenteur large de 16 bits a été placé au-dessus pour le progresser après chaque changement d'instruction, au moment où le contenu du CO stocké dans les latches du bus d'adresses externe était réintégré à sa place. Mais il suffit d'un coup de pouce pour qu'un incrémenteur puisse aussi devenir décrémenteur : dès lors, il pouvait servir à incrémenter *ou* décrémenter le pointeur de pile, permettant de réaliser les instructions de stockage et de déstockage sur pile. Enfin, n'importe quelle *paire* de registres peut être également incrémentée ou décrémentée sur les 16 bits : comme ils servent toujours à l'adressage mémoire pour certaines des instructions... on appréciera de ne plus affronter le problème de pagination pour les opérandes, rencontré avec le 8008, à cause de la séparation de H et L !

Voici les possibilités nouvelles les plus importantes que cette architecture - et d'autres détails que nous verrons plus loin - a apportées au 8080 :

— possibilité de stocker et de déstocker rapidement sur pile les contenus des trois paires de registres B-C, D-E, H-L et du couple ACCU-DRAPEAUX (8 + 5 bits de conditions). Les opérations de sauvetage état/registres machine à l'arrivée d'une interruption, et de restauration après service, sont ainsi réalisées rapidement. De plus, la pile externe peut servir de véritable extension au jeu de registres généraux ;
— on peut réaliser des additions en double précision entre H-L et l'un des autres registres ;
— l'instruction de *correction décimale* (decimal adjust) permet une arithmétique DCB à peu près aussi rapide que la binaire ;
— les nombreux signaux de commande et d'état (4 en entrée et 6 en sortie) simplifient l'interfaçage, permettant le DMA (*direct memory access*, dont nous parlerons beaucoup plus loin) et facilitent la mise au point des programmes.

3 - 7 - 3. Bons signaux d'état et de commande : atouts pour le système et le bus

Très précisément, ces signaux directement disponibles sur les broches du boîtier CPU sont les suivants (nous traduisons leur signification, mais par la suite nous garderons le libellé original, sinon nos lecteurs nous voueront aux gémonies lorsqu'ils se plongeront dans les brochures d'Intel pour faire cracher du feu à Alcyane...) :

— SYNC (synchronisation), en sortie. Signal de synchronisation indiquant le commencement de chaque cycle machine.
— DATA BUS INPUT (entrée bus données, souvent abrégé en DBIN), en sortie. Signal indiquant à quel moment le bus données est en mode réception (le CPU signale ainsi qu'il attend des données).
— READY (prêt, comme au tennis), en entrée. Signal indiquant au CPU qu'une donnée valide est disponible. S'il n'est pas donné, le CPU se met en état WAIT.
— WAIT (attente), en sortie. Signal indiquant que le CPU est dans l'état WAIT.
— WRITE (écrire), en sortie. Signal qui prévient la mémoire et les appareils de sortie qu'une donnée valide provenant du CPU est disponible sur le bus données. En vérité, c'est le signal WR

(not write) qui est sorti : quand il tombe à zéro, la donnée est stable sur le bus.

— HOLD (retenir), en entrée. Signal envoyé par un appareil externe pour demander accès aux bus adresses et données du CPU. La demande est accordée au moment où se termine l'accès mémoire, et l'accusé de réception est donné sur la broche de sortie HOLD ACKNOWLEDGE. Les bus adresses et données du CPU deviennent flottants, (dans un état à haute impédance (résistance si le terme vous gêne)) isolant le CPU du reste su système. A l'intérieur, le CPU termine l'exécution du cycle mémoire en cours. Ensuite, il reste au repos aussi longtemps que HOLD est actif. HOLD et HOLD ACKNOWLEDGE peuvent être utilisés pour réaliser un DMA (au cours duquel un périphérique accède à grande vitesse directement à la mémoire, sans passer par le CPU, sous le contrôle d'une logique hardware externe) ou dans les applications multiprocesseurs... eh oui, on peut faire ça aussi. A vous d'écrire le software mes jolis, si ça vous chante.

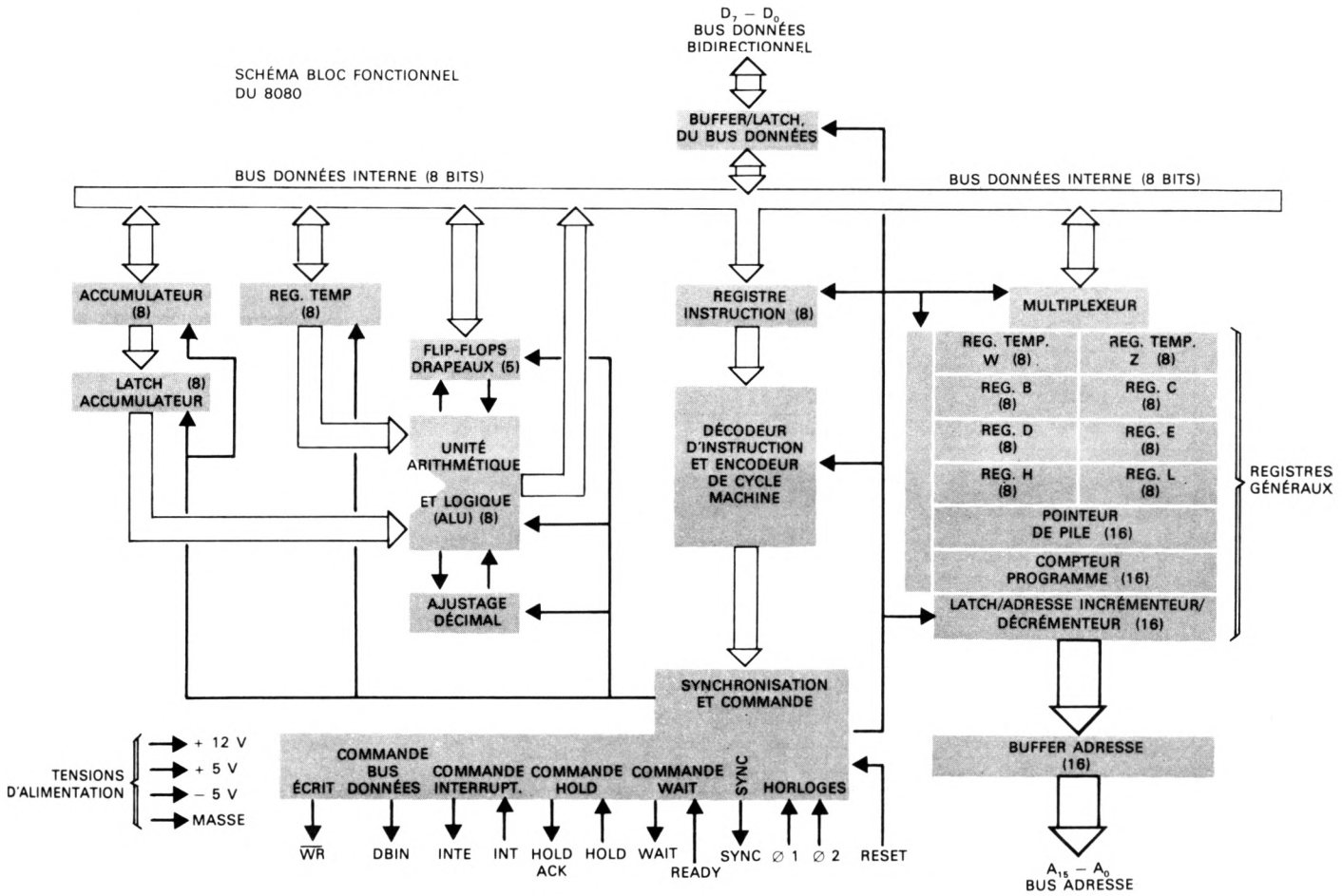
— HOLD ACKNOWLEDGE (accusé de réception du HOLD), en sortie. Indique que l'état HOLD est en cours. Abrégé en HLDA.

— INTERRUPT REQUEST (demande d'interruption), en entrée. La ligne d'entrée des interruptions est échantillonnée à la fin du cycle d'instruction en cours, et si la bascule interne de permission d'interruption a été mise *en* (opp. *hors*) par le software (angl. if the internal software control interrupt enable flip-flop is set ; on peut dire aussi : si les interruptions sont armées, autorisées, démasquées) la séquence de service de l'interruption est déclenchée. Abrégé en INT.

— RESET (remise à zéro), en entrée. Ligne qui permet de remettre à zéro le compteur ordinal de façon à ce que la première instruction de programme exécutée soit celle située à l'adresse zéro en mémoire.

— INTERRUPT ENABLE (autorisation d'interruption), en sortie. Signal indiquant l'état du flip-flop d'autorisation d'interruption. Abrégé en INTE.

Le processeur envoie également 8 bits d'état, engendrés par sa logique interne, sur le bus données à la période T_2, au moment du front descendant de SYNC (nous verrons que le bus n'a rien à faire à ce moment là) et nous les stockerons



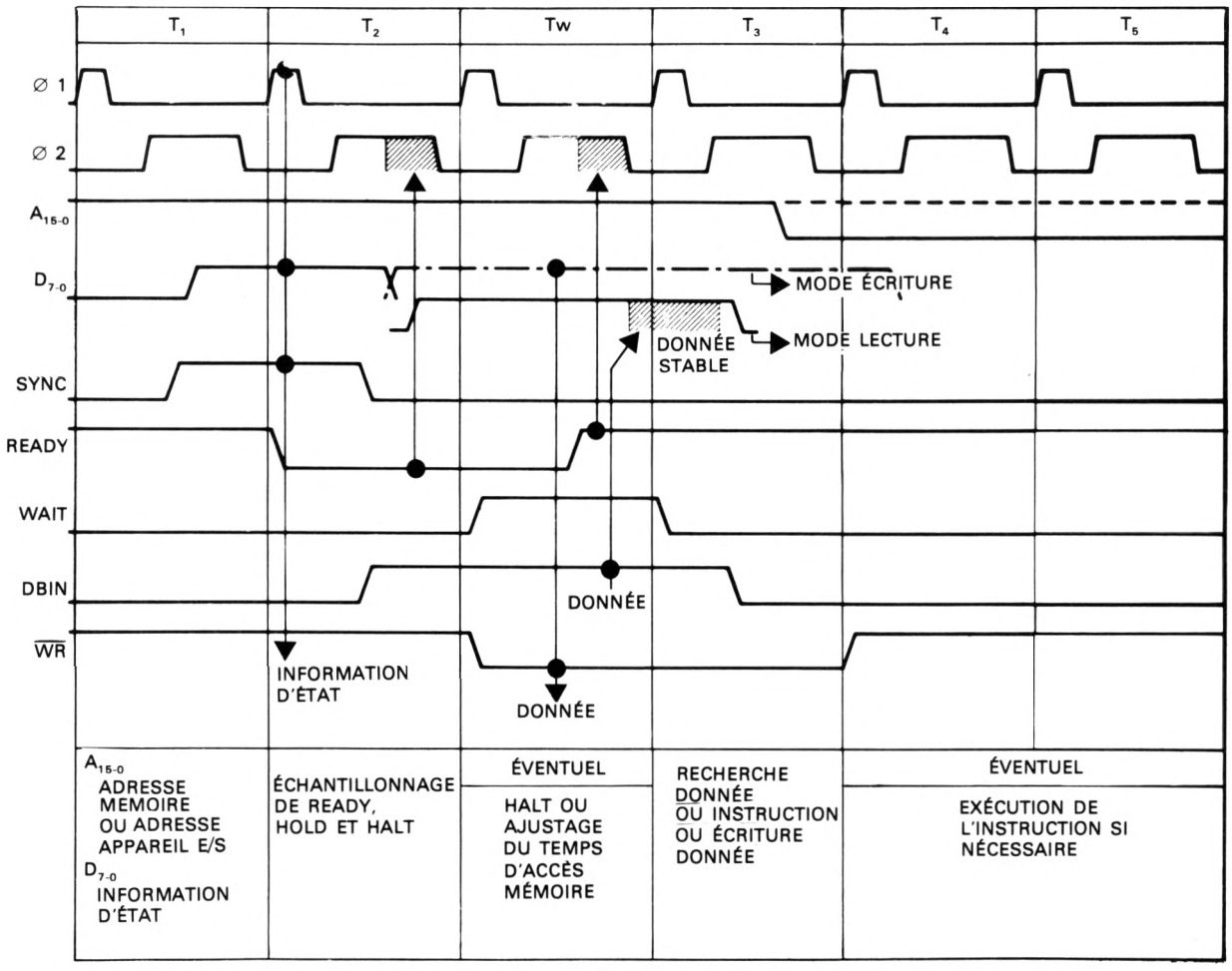
soigneusement dans un latch externe. Voici les 8 bits d'état en question :

— HALT ACKNOWLEDGE - réponse à l'instruction HALT. Abrégé en HLTA.
— INTERRUPT ACKNOWLEDGE - résulte de l'acceptation par le CPU d'une demande d'interruption. Abrégé en INTA.
— INPUT CYCLE (cycle d'entrée) - indique que le bus adresse détient l'adresse d'un appareil d'entrée : dès que DBIN sera actif la donnée devra être placée sur le bus de données. Abrégé en INP.
— OUTPUT CYCLE (cycle de sortie) - indique que le bus adresse détient l'adresse d'un appareil de sortie : la donnée sera sur le bus de données lorsque $\overline{WR}$ sera actif. Abrégé en OUT.
— MEMORY READ (lecture mémoire) - indique que le bus de données va recevoir une donnée provenant de la mémoire. Abrégé en MEMR.
— M_1 - indique que le cycle en cours sert à aller chercher le premier octet (*byte*) d'une instruction.
— STACK (pile) - indique que le bus adresse détient l'adresse du sommet de la pile externe.
— $\overline{WRITE\ OUTPUT}$ (non écrire sortie) - indique que le bus des données va disposer ou non d'une donnée devant être écrite en mémoire ou destinée à une opération de sortie. En fait, ce bit d'état correspond à $\overline{WO}$ (not writeoutput) et c'est lorsqu'il est par conséquent égal à zéro qu'il y aura écriture/sortie pendant le cycle machine en cours.

3 - 7 - 4. Une seconde de soixante kilomètres

Les festons chronologiques réguliers $\Phi 1$ et $\Phi 2$ des deux horloges rythment l'entrecroisement de signaux dans notre machine, grignotant fébrilement le travail à exécuter à quelques deux mégahertz... vitesse qui, il n'y a pas bien longtemps, paraissait fort respectable pour un ordinateur « lourd ».

Le déroulement complet d'une instruction demande un à cinq cycles machine selon sa com-



CHRONOGRAMME DES PRINCIPAUX SIGNAUX DU 8080

plexité. Les instructions sont de longueur variable, un, deux ou trois octets. Comme pendant un cycle machine on lit ou on écrit un octet, il s'ensuit que seules les instructions d'un octet pourront ne durer qu'un cycle machine, et que celles de deux et trois octets prendront au minimum respectivement deux et trois cycles machine.

Un cycle machine, lui, se divise en plusieurs microcycles durant chacun une période d'horloge, soit 0,5 μs quand l'horloge est réglée à deux MHz. Sur les chronogrammes, ces microcycles sont baptisés T_1, T_2, etc. et correspondent à des *états* successifs du CPU qui se succèdent dans un ordre constant au cours des cycles machines successifs M_1, M_2, etc. d'une instruction. Le cycle M_1 comprend quatre ou cinq états (microcycles) successifs : c'est pendant M_1 que le code opération de l'instruction à exécuter est lu en mémoire, quelle que soit la longueur de l'instruction. Les cycles suivants, M_2, M_3, M_4 et M_5 (s'il y a lieu) durent normalement chacun trois périodes d'horloge.

En plus des états T_1 à T_5, il existe trois autres états qui peuvent s'intercaler dans la séquence normale T_1-T_5, et durer un nombre de périodes d'horloge indéfini. En effet, ces états qui sont WAIT, HOLD et HALT, sont sous le contrôle de signaux extérieurs qui les déclenchent ou les arrêtent, nous verrons comment.

Nous allons maintenant suivre en détail le mode de fonctionnement du 8080 en nous référant simultanément à deux figures, le chronogramme décomposant dans le temps le cycle machine, et le diagramme d'état du CPU indiquant les conditions de transition d'un état T à un autre état T.

Si une instruction vient juste de s'achever, et que notre cycle machine est donc le premier d'une nouvelle instruction, il débute par T_1 pendant lequel le contenu du PC (20) est envoyé sur le bus adresse afin de permettre de rechercher un peu plus tard en mémoire le code opération de l'instruction à exécuter (ce code opération occupe un octet entier) ; SYNC est vrai ; le bus données D, qui n'a encore rien à faire à ce moment là, reçoit en provenance du CPU nos fameux huit bits d'état, correspondant à la nature du cycle machine qui est en train de débuter.

Dans notre cas précis, nous aurons bien sûr $M_1 = 1$, MEMR = 1 et $\overline{WO} = 1$, les cinq autres bits étant à zéro comme vous pourrez le vérifier d'après les définitions de INTA, STACK, HLTA, OUT et INP. On utilise, à l'aide d'un fil relié à une sortie de l'horloge, le front descendant de *O*1 comme signal pour *gater* les huit bits dans un latch externe d'état relié au bus, les mettant soigneusement à l'abri (par gater, affreux terme, nous voulons dire non pas pourrir ces pauvres bits, mais ouvrir la porte, gate, qui leur permettra d'entrer dans les bascules à verrouillage, et clac...

(20) A partir d'ici, pour ne pas semer la confusion chez les lecteurs qui se réfèreraient aux brochures américaines du 8080, au lieu de CO (compteur ordinal), ou de RAI (registre adresse instruction), nous utiliserons PC (program counter), expressions ayant la même signification : elles désignent toutes le registre où se trouve en permanence l'adresse en mémoire centrale du prochain octet d'instruction à aller chercher.

TABLEAU DES CONFIGURATIONS DU MOT D'ÉTAT

TYPE DE CYCLE MACHINE

BIT DU BUS DONNÉE	INFORMATION D'ÉTAT	RECHERCHE INSTRUCTION	LECTURE MÉMOIRE	ÉCRITURE MÉMOIRE	LECTURE PILE	ÉCRITURE PILE	LECTURE ENTRÉE	ÉCRITURE SORTIE	ACCUSÉ RÉCEPT. INTERRUPT.	ACCUSÉ RÉCEPT. HALTE	ACCUSÉ RÉCEPT. INTERRUPT. PENDANT HALTE
		①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩ NUMÉRO DU MOT D'ÉTAT
D_0	INTA*	0	0	0	0	0	0	0	1	0	1
D_1	$\overline{WO}$	1	1	0	1	0	1	0	1	1	0
D_2	STACK	0	0	0	1	1	0	0	0	0	0
D_3	HLTA	0	0	0	0	0	0	0	0	1	1
D_4	OUT	0	0	0	0	0	0	1	0	0	0
D_5	M_1	1	0	0	0	0	0	0	1	0	1
D_6	INP*	0	0	0	0	0	1	0	0	0	0
D_7	MEMR*	1	1	0	1	0	0	0	0	1	0

on aurait pu dire *strober*, aussi affreux, un strobe étant en anglais une impulsion qui ouvre au bon moment une porte, allume un digit, etc.).

T_2 suit toujours T_1 (on le voit sur le diagramme d'états), et pendant T_2 les signaux READY, HOLD et HALT sont testés. Si READY est vrai, on peut entrer dans l'état T_3 ; sinon, le CPU se met en état d'hibernation wait (Tw) et y restera aussi longtemps que READY sera faux.

Le signal READY est particulièrement précieux, car il permet de synchroniser le CPU avec des événements extérieurs. Donnons-en deux exemples :

— si la mémoire qui vient d'être adressée signale qu'elle n'est pas prête à délivrer son information, on va l'attendre. Il est donc non seulement possible d'utiliser des mémoires *aussi lentes qu'on le veut*, mais aussi qu'elles *coexistent* avec d'autres plus rapides,
— en installant derrière la face avant d'Alcyane une quantité suffisante de boîtiers DIL, il sera possible de commander manuellement à travers eux le READY pour avancer en pas à pas à travers le programme, d'un cycle machine à chaque fois, en s'arrêtant à la fin de chaque état T_2. Les contenus des bus adresse et données, ainsi que certaines combinaisons des signaux d'états, seront alors affichés par les LEDs (diodes électro-luminescentes) du pupitre, et l'on pourra suivre en détail le déroulement des instructions, pour l'enregistrement ou pour faciliter la mise au point de parties délicates de programmes. Le rôle des DIL dans cette affaire sera de relever READY rapidement, et de le rabaisser de même avant le T_2 suivant, ce qui serait plutôt difficile à réaliser directement à la main !

Pendant T_3 l'octet venant de la mémoire est disponible sur le bus, et comme nous sommes en M_1 il va être transféré dans le registre instruction, puisqu'il s'agit du code opération de l'instruction qu'on est en train d'amener au CPU. Dans un cycle machine autre que M_1, l'octet situé sur le bus données subirait un autre sort : être transféré dans un autre registre, ou même circuler dans l'autre sens, c'est-à-dire être écrit. Le chronogramme indique simultanément deux de ces possibilités (incompatibles !). Dans l'une on a $\overline{WR}$ faux (= 0), donc WR vrai, c'est-à-dire écriture, la donnée étant sur le bus D comme l'indique le trait mixte. Si au contraire on a DBIN vrai, comme vu il signale qu'on est en mode lecture et commande à la mémoire d'amener sur le bus la donnée. Comme ici READY est redevenu haut (vrai) rapidement, Tw n'a duré qu'un microcycle, T_3 a pu reprendre dans la foulée et la donnée étant stable sur le bus à ce moment (zone hachurée) est entrée dans le CPU. C'est le décodeur d'instructions qui, chargé maintenant, va la décoder et activer les circuits de synchronisation et de commande, envoyant sur la surface du chip les signaux qui contrôlent les transferts de données internes et déclenchent l'enchaînement des cycles machine (voir schéma bloc fonctionnel du 8080).

A la fin de T_4 si le cycle machine est terminé, sinon à la fin de T_5, le 8080 revient à l'état T_1 pour débuter un nouveau cycle machine : si le déroulement de l'instruction n'est pas achevé, ce sera un cycle M_2, qui pourra être suivi de M_3, M_4 et même M_5 pour les instructions les plus compliquées (nous en détaillerons une...). Ces cycles, répétons-le, ne durent normalement, eux, que trois périodes chacun.

En consultant le diagramme d'états, on remarque que c'est seulement pendant le dernier état du dernier cycle machine d'une instruction que la ligne de demande d'interruption est testée, ceci de façon à n'accepter une interruption qu'à la fin de l'exécution d'une instruction. Du reste, c'est le produit logique de INT et INTE qui est testé, c'est-à-dire que l'interruption n'est acceptée que si d'une part elle est demandée, et si d'autre part la bascule interne de permission d'interruption (INT F/F, flip-flop) a été armée par le programme à l'aide de l'instruction EI (*Enable Interrupts*) *au moins deux instructions auparavant.* Cette restriction, câblée dans le CPU, sert à permettre l'exécution complète d'une instruction RET de retour de sous-programme dans lequel les interruptions ont été démasquées juste avant de revenir au programme principal (ou à un sous-programme appelant de niveau supérieur), lequel aura alors le temps si désiré de masquer à nouveau les interruptions par DI (*Disable Interrupts*).

Si donc INT.INTE = 0, on entre dans un cycle M_1 spécial pendant lequel le contenu de PC, contrairement à la normale, n'est pas incrémenté après avoir été mis dans les latches du bus adresse, au moment de son retour dans le registre PC. Les huit bits d'état sont alors sortis, et bien sûr INTA = 1, donnant l'accusé de réception de l'interruption. Pendant ce cycle, l'appareil qui a provoqué l'interruption enverra au CPU l'une des huit instruction RST possibles *(restart)* dont nous détaillerons le fonctionnement précis plus tard.

Comme on le voit, le nombre d'états requis pour l'exécution d'une instruction va d'un minimum de quatre pour les instructions ne référençant pas la mémoire, telles que les instructions entre registres ou les instructions arithmétiques de l'accumulateur, à un maximum de 18 pour les instructions les plus complexes (comme XTHL, qui échange le contenu des registres H et L avec celui des deux positions supérieures de la pile externe). Si l'horloge fonctionne à la fréquence maximum permise du 8080A normal, deux mégahertz, chaque état durant alors 500 nanosecondes, on voit que les durées d'exécution des instructions iront de deux à neuf microsecondes.

Enfin, si une instruction HALT s'exécute, le processeur entre dans l'état WAIT dont il ne pourra cette fois sortir que si une interruption se manifeste : on voit là l'utilité particulière de l'instruction HALT. Cependant, si nous arrêtons ainsi un programme, le pupitre d'Alcyane va devenir inerte. Pour stopper sans se verrouiller de cette manière, il vaudra donc mieux utiliser le vieux truc de l'instruction de branchement se rebouclant sur elle-même comme un serpent qui se mord la queue. Les lumières de la console deviendront fixes, alertant l'utilisateur, et en passant en MANUEL (abaissement de READY), on pourra reprendre le contrôle des opérations au pupitre avec des commandes qui réagissent.

Au fil des explications, nous avons -peut-être presque sans nous en apercevoir - fait un plongeon dans un monde étrange, microscopique dans le temps et dans l'espace, où les signaux s'entrecroisent à un rythme hallucinant, $\Phi 1$ et $\Phi 2$ grignotent le chronogramme de façon obstinée et rageuse, à une vitesse dont on se fait difficilement une idée. De là notre titre de paragraphe, car nous avons eu un choc après avoir dessiné notre premier chronogramme de signaux, pour étudier l'horloge d'Alcyane, en découvrant qu'à trois centimètres de papier par période le chronogramme complet d'une seconde s'étalerait sur toutes les pages de 5 000 numéros de *l'Informatique nouvelle* !

3 - 7 - 5. Mot d'état : une fenêtre sur le CPU

L'un des problèmes que pose un microprocesseur par rapport à un ordinateur traditionnel est dû au revers de sa médaille - dont fort heureusement l'endroit a beaucoup plus d'importance ! Dans le gros ordinateur, l'unité centrale s'étend sur plusieurs mètres ; jadis, on pouvait même y pénétrer par une porte, et l'on parlait de «walk-in processor»... Il est alors très facile d'accéder aux divers éléments du processeur : registres, points de diagnostics, etc. Dans un μP, minuscule boîte fermée, se passent une foule d'événements dont seuls les résultats finaux parviennent au monde extérieur par quelques dizaines de broches. Fort heureusement, dans notre 8080, il existe ces fameux huit bits sortis en T_1 T_2, et qui sont d'un grand secours. A chaque cycle machine, tel un chef d'orchestre déléguant des messagers à ses exécutants, le CPU signale ses états d'âme sous forme d'un véritable *mot d'état du processeur* prévenant en détail de ce qui va se passer et des actions à effectuer dans le proche avenir.

Le tableau des configurations du mot d'état mérite d'être examiné un moment. Tout d'abord, pris individuellement, les divers bits indiquent huit conditions qui sont chacune exploitables pour sélecter les éléments du système autour du CPU qui devront être actifs. Mais en les combinant logiquement, on peut obtenir des signaux plus synthétiques si on le désire. Notons en particulier que les trois bits repérés d'un astérisque permettent de commander complètement le trafic des données sur le bus données du 8080. INTA par exemple, sera utilisé pour ouvrir la moitié de la porte (si l'on ose dire !) à une instruction RESTART câblée extérieurement qui sera entonnée sur le bus DATA au moment où DBIN devenu actif finira de l'ouvrir (ce signal lui, rappelons-le, contrairement aux bits d'état fugitifs qu'il faut latcher, fait partie de la famille de ceux qui sont directement disponibles au moment voulu sur les broches du μP, comme par exemple $\overline{WR}$ ou HLDA).

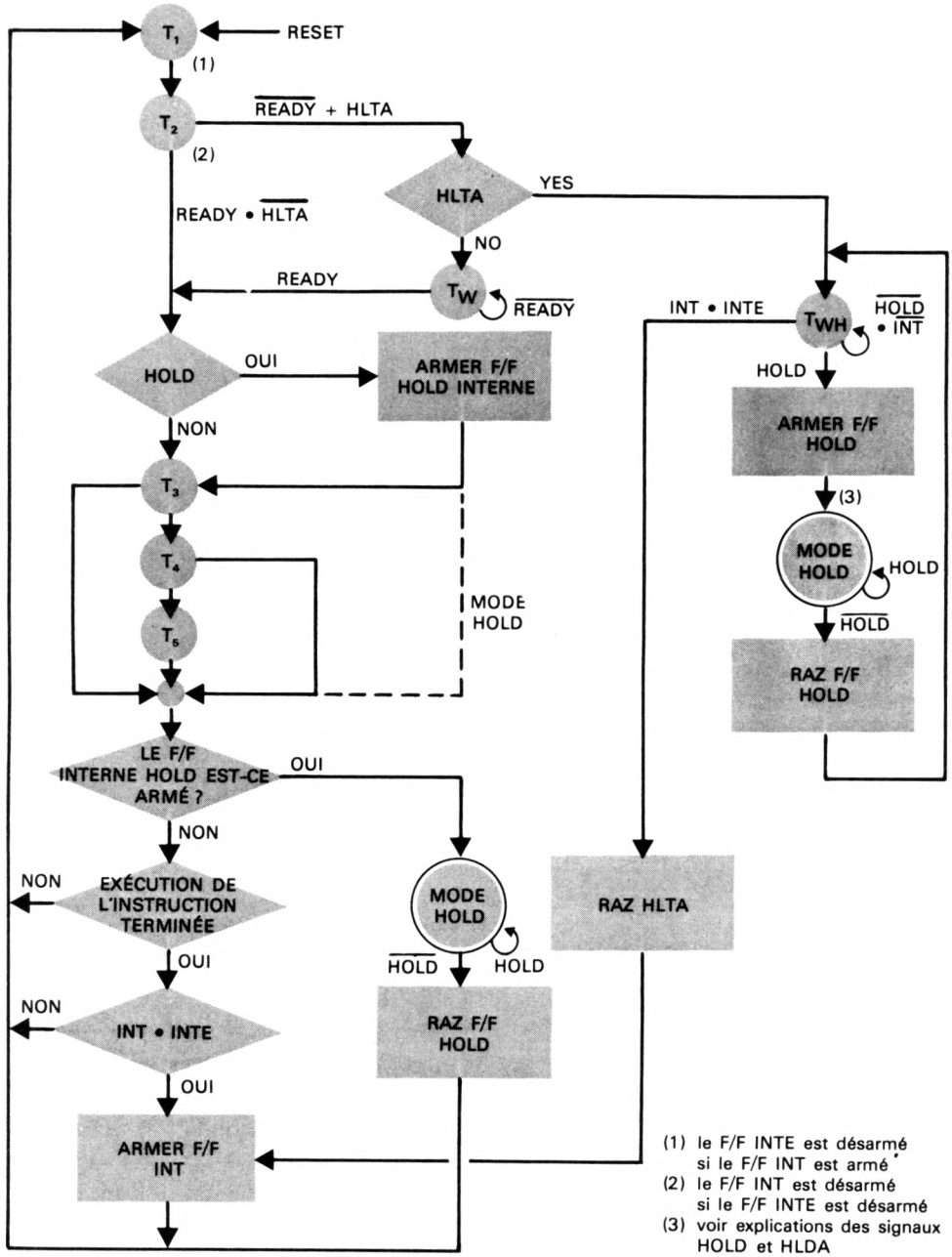
Prise en bloc, la configuration des huit bits identifie la nature du cycle machine M qui est en train de débuter : on voit qu'il y en a dix types différents, dont plusieurs ont déjà été vus. Seules ces dix configurations sont valides, sur les 256 possibles au total. On tient là, outre les possibilités d'action directe sur le système autour du CPU, un excellent moyen de diagnostic de défaillances qui permet d'y voir clair, au moyen d'un oscilloscope ou tout simplement de voyants en fonctionnant pas à pas.

3 - 7 - 6. Une instruction en pas à pas

Sur le point de quitter pour un bon moment l'électronique en passant à la programmation, nous allons ménager une transition entre les deux en suivant pas à pas, exactement comme nous pourrons le faire au pupitre avec les clés et les voyants, le déroulement d'une instruction parmi les plus complexes de la machine. Cette instruction, qui est l'une des plus puissantes et des plus régulièrement utilisées en programmation pour sa commodité, est l'instruction CALL d'appel de sous-programme avec préparation du retour automatique (en fin de sous-programme par l'une des neuf instructions de retour).

CALL, instruction de trois octets, demande cinq cycles machine pour son exécution, au cours de laquelle il va d'abord stocker en mémoire vive le contenu du PC. Ce stockage s'effectue avec enfoncement automatique de pile, et mise à jour du pointeur correspondant : il est ainsi possible d'utiliser des sous-programmes imbriqués, s'appelant mutuellement, pratiquement sans limitation de profondeur. Enfin, l'adresse de début du sous-programme est chargée dans le PC pour préparer le branchement...

Voyons maintenant cycle par cycle le déroulement des opérations, ce qui constituera une bonne récapitulation des notions fraîchement acquises. Tout commence naturellement par un cycle M_1 classique, où en T_1 le contenu de PC est



mis sur le bus adresse : c'est le cycle *fetch*, la *recherche* en mémoire du premier octet de l'instruction. A ce moment, le CPU ne «sait» encore pas que c'est un CALL qui va se présenter. Le contenu de PC, momentanément stocké dans les buffers/latches du bus adresse, sera retourné ensuite à travers l'incrémenteur dans le registre PC, augmenté d'une unité, pour pouvoir tout à l'heure aller chercher l'octet suivant de l'instruction. Au début de M_1/T_2, les huit bits d'état sont présentés par le CPU sur le bus données, et nous utiliseront la sortie d'une porte ET alimentée par SYNC et $\Phi 1$ pour les strober dans un latch externe de huit bits. Parmi les bits d'état, nous pouvons trouver sur le tableau que trois vont être «vrais», c'est-à-dire à un : $\overline{WO}$, M_1 et MEMR. Le premier indique qu'il n'y aura pas d'écriture en sortie du CPU : ce sera même en fait ici une lecture, donc une entrée dans ledit CPU. Le second indique bien que nous sommes en cours d'un cycle M_1 (un cycle normal ici, rappelons qu'il y en a un spécial en cas d'interruption). Le troisième indique enfin que le bus données va être utilisé au cours de ce cycle machine pour une lecture en mémoire, ce qui est bien compatible avec le reste. Tous ces signaux peuvent être utilisés dans les circuits pour les usages les plus divers.

En M_1/T_3, notre bit MEMR (stocké dans sa bascule à verrouillage) qui est relié à une porte ET où arrive également DBIN, va commander l'accès mémoire au bus DATA : le premier octet de l'instruction, le code opération donc, entre dans le CPU qui consacre les deux microcycles suivants M_1/T_4 et M_1/T_5 à décoder son contenu particulièrement complexe et à l'utiliser pour commander le positionnement de nombreuses portes sur le chip.

En M_2/T_1, début d'un nouveau cycle machine, le contenu précédent du PC, déjà augmenté de 1, est envoyé sur le bus adresse pour la recherche de l'octet suivant de l'instruction CALL. En M_2/T_2, toujours nos trois tests, et toujours SYNC et $\Phi 1$ qui stockent de nouveau dans le latch d'état du système le mot d'état présent de D_7 à D_0, les huit fils du bus DATA : il aura la même configuration qu'en M_1/T_1, à l'exception de M_1 qui ne sera plus vrai cette fois, bien évidemment. A partir de maintenant d'ailleurs, le mot d'état peut refléter la nature de l'instruction en cours, ce qui n'était pas possible en M_1/T_1 puisque le code opération n'était pas encore entré dans le processeur. Les lignes READY, HOLD et HALT sont testées, et si tout va bien on passe incontinent à M_2/T_3, où le deuxième octet de CALL va être chargé dans le registre temporaire Z faisant partie des registres généraux du CPU (voir schéma bloc). Le cycle M_2 est alors terminé : rappelons-nous en effet que tous les cycles autres que M_1 ne comprennent que trois états.

Le cycle M_3 démarre, et va être similaire à M_2 : le troisième octet du CALL est chargé dans le registre W. La paire W et Z contient ainsi l'adresse de la première instruction du sous-programme vers lequel on se branchera tout à l'heure. Mais il y a encore du travail à faire !

En M_4/T_1, c'est le contenu du pointeur de pile (SP, stack pointer) qui est mis cette fois sur le latch adresse et le bus A, et non pas celui du PC comme jusqu'ici. Cette adresse indique le premier emplacement libre de la pile, où va être stocké justement le contenu de PC, en tant qu'adresse de retour. N'oublions pas qu'à chaque fois que ce contenu est remis dans PC, il passe dans l'incrémenteur : donc présentement il est égal à la valeur initiale plus trois, c'est-à-dire qu'il pointe sur l'octet qui suit immédiatement l'instruction CALL, et c'est là que nous reviendrons donc après exécution du sous-programme. An M_4/T_2, tests et stockage du mot d'état. En M_4/T_3, la moitié de PC, c'est-à-dire un octet, sort sur D et est stockée à l'adresse de A, tandis que le contenu de SP revient du latch adresse et est *décrémenté* de un au passage, avant rangement dans SP.

Enfin M_5/T_1 : nous débutons le dernier cycle machine ; c'est encore le contenu de SP qui est mis sur A. Il pointe cette fois un cran plus haut, la pile se développant donc vers les adresses faibles. Après T_2 en T_3 le deuxième octet de PC est logé dans la pile, et tout est donc prêt pour le retour où l'instruction RET, incrémentant SP, récupérera en sens inverse les deux octets de la pile pour les charger dans PC. Il reste cependant encore une opération importante à assurer, le branchement vers le sous-programme : à première vue, il va falloir transférer WZ vers PC, restera-t-il le temps pendant T_3 ? C'est Federico Faggin - le concepteur du 8080, dont nous avons déjà parlé - qui donne la réponse (vous ne la trouverez pas dans les manuels !). En fait, ce transfert ne s'exécute pas, par suite d'une astuce digne des magiciens dont l'un des «trucs» principaux consiste à opérer avant ou après le moment où on les surveille...

Tout simplement, quelle que soit l'instruction qui dans la mémoire suit le call, pendans le cycle M_1 qui débute, c'est le contenu de WZ qui est envoyé sur le bus adresse ! La machine, dont le PC avait pourtant avancé normalement d'un cran, se retrouve alors avec le premier octet de la première instruction du sous-programme chargé dans son registre instruction, par un véritable tour de passe-passe... Quant au contenu de WZ, au retour du latch adresse il traverse bien normalement l'incrémenteur, reçoit son + 1 et est stocké dans PC : le saut est fait, on est sur de nouveaux rails.

Deux points pour finir : le programmeur peut choisir à n'importe quel moment de démarrer une nouvelle pile où il veut. Il dispose pour cela de l'instruction LXI SP que nous étudierons. Enfin, nous avons vu que le mot d'état du processeur comprend le bit STACK : on peut bien entendu l'utiliser en hardware, par exemple pour sélectionner systématiquement une carte mémoire particulière où l'on désire loger la pile pour la protéger.

3 - 7 - 7. Format des instructions et des données

Dans le 8080, l'unité manipulée est l'octet. Les données sont stockées dans chaque octet sous la forme d'entiers binaires de huit bits, et transférées telles quelles octet par octet sur le bus données. A l'intérieur d'un octet, le bit de droite (le moins significatif) aura comme indice zéro, et celui de gauche 7. Ainsi, A_2 sera le troisième bit de l'accumulateur en partant de la droite.

Les instructions ont un, deux ou trois octets de long, selon leur type :

— *Un octet*, qui est le code opération : instructions de registre à registre, opérations en mémoire (adressage indirect), arithmétique, logique, décalages, retour, gestion de la pile et des interruptions.
— *Deux octets*, le second étant soit une opérande immédiat soit l'adresse d'un appareil d'entrée

ou de sortie (on peut donc en avoir 256 en entrée, autant en sortie).

— *Trois octets*, le deuxième (*attention* !) contenant la moitié de *faible poids* de l'adresse opérande (low address), et le troisième la moitié de *poids fort* (référez-vous aux paragraphes expliquant les cycles M_2 et M_3 de CALL, l'explication y est) : branchements, calls, chargement et stockage, avec adressage direct de 16 bits (64 Ko).

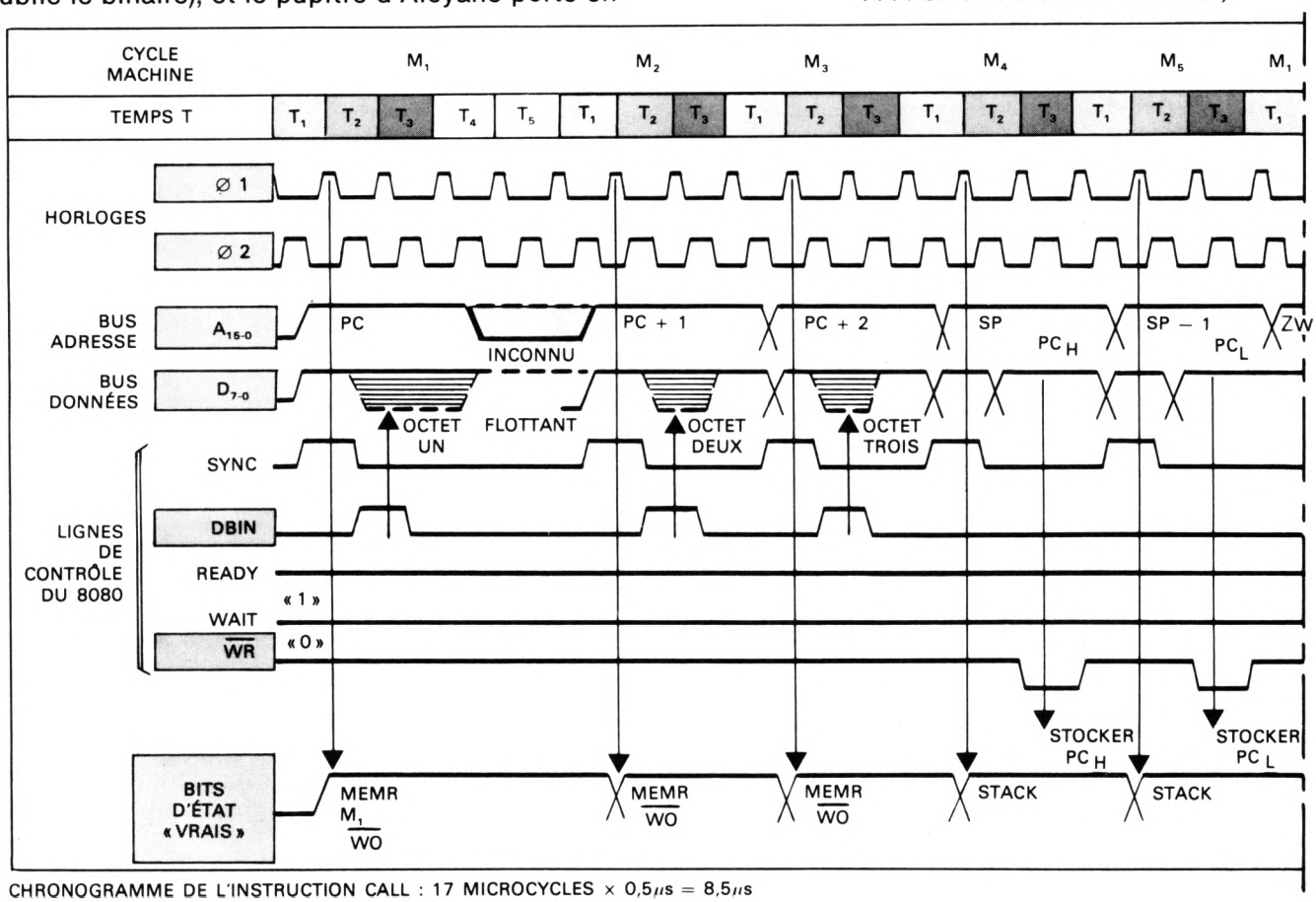
Il existe enfin un PSW (program status word), ou *mot d'état du programme*, qui comprend les huit bits de l'accumulateur et les quatre drapeaux de condition. Ne pas confondre avec le mot d'état du processeur !

3 - 8. LES INSTRUCTIONS DU 8080

3 - 8 - 1. Hexa et octal

La structure de base de l'information dans le 8080 étant l'octet, on peut difficilement échapper à l'hexadécimal pour la représenter commodément. Certes, on pourrait utiliser l'octal (relisez «*Un Fil d'Ariane : l'ordinateur à la portée de tous*» si ces termes vous laissent rêveur, ou si vous avez oublié le binaire), et le pupitre d'Alcyane porte en fait les deux types de séparation, 3 bits par 3 bits en octal au-dessus des voyants, et 4 par 4 en hexa au-dessous. Vous avez le choix. Cependant, sachez que toute la littérature concernant les machines à huit bits utilise l'hexa, et c'est la notation que nous vous conseillons d'employer, pour ne pas vous embrouiller ; mais si vous êtes un fana de l'octal (comme beaucoup d'étudiants américains) aucune impossibilité technique. Rappelons les 16 chiffres hexadécimaux équivalant aux configurations binaires :

0000 binaire s'écrit 0 en hexa,
0001 binaire s'écrit 1 en hexa,
0010 binaire s'écrit 2 en hexa,
0011 binaire s'écrit 3 en hexa,
0100 binaire s'écrit 4 en hexa,
0101 binaire s'écrit 5 en hexa,
0110 binaire s'écrit 6 en hexa,
0111 binaire s'écrit 7 en hexa,
1000 binaire s'écrit 8 en hexa,
1001 binaire s'écrit 9 en hexa,
1010 binaire s'écrit A en hexa,
1011 binaire s'écrit B en hexa,
1100 binaire s'écrit C en hexa,
1101 binaire s'écrit D en hexa,
1110 binaire s'écrit E en hexa,
1111 binaire s'écrit F en hexa,



CHRONOGRAMME DE L'INSTRUCTION CALL : 17 MICROCYCLES × 0,5 µs = 8,5 µs

Pour nous conformer aux usages établis (et en particulier à la notation utilisée par la langage d'assemblage, ou disons *l'assembleur* du 8080), nous ferons suivre les information en hexadécimal de la lettre H accolée, celles en décimal de D et celles en binaire de B. Ainsi, on pourra écrire :

86H = 10000110B = 134D

3 - 8 - 2. Un peu d'arithmétique

Les instructions arithmétiques du 8080 permettent de travailler en décimal (décimal codé binaire classique à quatre bits par chiffre décimal, soit deux chiffres décimaux par octet) ou en binaire. Un mot sur les conventions utilisées dans ce dernier cas.

Il existe plusieurs méthodes de représentation des nombres tenant compte des nombres négatifs et des signes : celle utilisée par le 8080 est la plus pratique et la plus courante, le format en *complément à deux*, sur lequel on travaille en *arithmétique en complément à deux*, et c'est bien sûr cette dernière qui est câblée sur le chip μP. En effet, une opération de soustraction dans ce format se réduit simplement, comme on va le voir, à une séquence de complémentations et d'additions, ce qui est compatible avec la simplicité recherchée pour le dessin du chip. Notons d'ailleurs que c'est le même système qui est employé jusque dans les plus gros ordinateurs...

Pour comprendre comment on peut représenter des nombres négatifs de cette façon dans un ordinateur, considérons un registre mécanique tel que le totaliseur kilométrique d'une voiture par exemple. Si nous faisons tourner à l'envers ce compteur de cinq chiffres en approchant de zéro et en le franchissant nous verrons successivement :

00005
00004
00003
00002
00001
00000
99999
99998
etc.

Il est clair que le nombre 99998 correspond à — 2. De plus, si nous faisons l'addition

00005
99998

Report 1 00003

en ignorant le report qui sort à gauche, nous avons en réalité effectué la soustraction

5 — 2 = 3

Le nombre 99998 de cet exemple, est en fait le *complément à dix* de 2. Ainsi, en système décimal, on peut remplacer la soustraction par l'addition du complément à dix du nombre à soustraire.

Si l'on utilise le système de complément pour représenter les nombres négatifs, on pourra se passer du signe moins pour ces nombres négatifs. Alors tous les nombres pourront être représentés par cinq chiffres dans notre ordinateur : 2 par 00002 et 99998. Le fait d'utiliser un tel système demande qu'une convention soit établie afin de définir ce qui est et ce qui n'est pas un nombre négatif. Si, par exemple, nous faisons tourner notre compteur à l'envers jusqu'à 48732, un observateur arrivant à ce moment-là ne saura pas si ce que représente le compteur est le nombre négatif 51268 ou le nombre positif 48732. Cinq chiffres donnant la possibilité de représenter 100 000 nombres (de 0 à 99999) il semble raisonnable de les partager également entre nombres positifs et nombres négatifs. Dans ce cas, ceux de 0 à 49999 seront considérés comme positifs, et ceux de 50000 à 99999 comme négatifs : un chiffre à gauche supérieur à 4 sera l'équivalent d'un signe moins, nous travaillons en *complément à dix signé*.

Tout ceci peut se transposer du décimal en binaire, en remplaçant le mot *dix* par *deux*.

Lorsqu'on regarde un octet (qu'on sait d'avance représenter un nombre en *complément à deux signé,* sinon ce peut être aussi bien un signe graphique, un code opération d'instruction ou une configuration de bits destinée à faire avancer un moteur pas à pas) les sept bits de droite représentent la valeur du nombre, et le bit de gauche son signe : 0 pour un nombre positif, 1 pour un nombre négatif. L'étendue des nombres positfs qu'on peut représenter dans un octet en complément à deux signé va donc de 0 à 127 D :

0D = 00000000B = 0H
1D = 00000001B = 1H
. . . . . .
. . . . . .
126D = 01111110B = 7EH
127D = 01111111B = 7FH

Nous constatons bien que le bit de gauche est toujours zéro.

Pour changer de signe un nombre en complément à deux, on applique les règles suivantes :

a) Complémenter chaque bit du nombre (c'est-à-dire remplacer les 1 par des zéros et les zéros par des 1) : on obtient ainsi le *complément à un.*
b) Ajouter un au résultat (au bit de droite), sans s'occuper d'une éventuelle retenue s'échappant à gauche du nombre.

Exemple : nous voulons écrire —10D en complément à deux. Ecrivons d'abord + 10D : ce sera

00001010B. Complémentons chaque bit : 11110101B. Ajoutons 1 : il vient 11110110B qui est le résultat cherché ; on peut l'écrire F6H en *sténo hexa*, puisque l'hexa sera surtout pour nous une sténo du binaire, condensé quatre fois. Remarquons que le bit de gauche est un 1, c'est-à-dire un signe moins.

Exemple : quelle est la valeur de 86H interprété en tant que nombre signé en complément à 2 ? Le bit de gauche est un, il s'agit donc d'un nombre négatif. Pour obtenir sa valeur absolue, complémentons chaque bit et ajoutons 1 :

86 H = 10000110B
complémentons : 01111001B
ajoutons 1 : 01111010B

La valeur absolue est 7AH = 122D. Notre nombre vaut donc —122D.

L'étendue des nombres négatifs représentables va de —1 à —128D.

—1D = 11111111B = FFH
—2D = 11111110B = FEH
.....
.....
—127D = 10000001B = 81H
—128D = 10000000B = 80H

Nous constatons bien que le bit de gauche est toujours 1.

Nous allons maintenant effectuer la soustraction 1AH — 0CH par addition, ce qui nous amène à réaliser les opérations suivantes :

Formons le complément à deux de 0CH : c'est F4H (le complément de F est 0, de E est 1, de D est 2, de C est 3, et n'oublions pas d'ajouter 1). *Ajoutons-le* au premier nombre :

1AH = 00011010B
+(—0CH) = F4H = 11110100B

00001110B = OEH, réponse correcte

Lorsqu'un octet est interprété comme nombre en complément à deux *non signé*, sa valeur est systématiquement considérée comme positive et dans l'étendue 0 à 255D :

0 = 00000000B = 0H
1 = 00000001B = 1H
.....
.....
127D = 01111111B = 7FH
128D = 10000000B = 80H
.....
.....
255D = 11111111B = FFH

A première vue, ceci étant la représentation binaire la plus normale qui soit, on se demande ce que signifie la dénomination précédente. C'est qu'en fait l'arithmétique en complément à 2 est encore valide ici.

Lorsqu'on effectue une instruction d'addition, le bit Carry (report) situé dans l'ALU est mis à un quand le résultat dépasse 255D, pour prévenir donc des dépassements de capacité.

Lorsqu'on effectue une instruction de soustraction, le bit C est remis à zéro quand le résultat est positif. Si le bit C est mis à un, c'est que le résultat est négatif et se présente sous forme de complément à 2. *Ainsi, le bit Carry lorsqu'il est à un indique la présence d'une retenue soustractive* (borrow).

Exemple 1 : soustraire 98D de 197D par une instruction de soustraction en utilisant l'arithmétique en complément à 2 non signée. Dans la machine l'opération est réalisée comme vu par complémentation et addition :

197D = 11000101B = C5H
—98D = 10011110B = 9EH

retenue 1 01100011B = 63H = 99D

Comme la retenue en provenance du bit 7 (extrême gauche) indique que le résultat est en clair et positif, les circuits de soustraction vont remettre à zéro le bit Carry.

Exemple 2 : soustraire de même 15D de 12D en utilisant l'arithmétique en complément à 2 non signée :

12D = 00001100 = OCH
—15D = 11110001 = F1H

retenue 0 11111101 = —3D

Comme la retenue en provenance du bit 7 est zéro, ceci montre que le résultat est négatif et sous forme complémentaire à 2, et le bit C sera mis à 1 par les circuits de soustraction, indiquant qu'une retenue soustractive s'est produite.

Les instructions du 8080 utilisant l'opération de soustractions sont SUB, SUI, SSB, SBI, CMP et CMI. Bien que le même résultat puisse être obtenu par addition d'un nombre préalablement complémenté ou par soustraction d'un nombre en clair, le bit C aura selon le cas une valeur différente. Les instructions utilisant l'opération d'addition sont ADD, ADC, ADI, ACI et DAD.

Exemple : si l'on obtient le résultat —3D en effectuant ADD de +12D et —15D, le bit C sera mis à zéro ; si le même résultat est obtenu par SUB entre +12D et +15D, le bit C sera mis à 1. Les deux instructions indiquent ainsi chacune à leur manière que le résultat est négatif : il faut être bien aver-

ti, quand on programme, des opérations qui mettent C à 1 et de celles qui le mettent à zéro :

ADD + 12D et —15D :

+ 12D = 00001100B
+ (—15D) = 11110001B

retenue 0 11111101B = —3D

provoque la mise à zéro de C (il n'y a pas eu de dépassement de capacité).

SUB + 15D de + 12D :

+ 12D = 00001100B
—(+ 15D) = 11110001B

retenue 0 11111101B = —3D

provoque la mise à un de C.

3 - 8 - 3. Symbolisme des instructions

Nous allons définir maintenant le symbolisme simple utilisé pour décrire le rôle des instructions que peut exécuter notre μP.

— Le premier octet sera représenté par un code opération symbolique en plusieurs lettres, majuscules et minuscules avec éventuels indices.

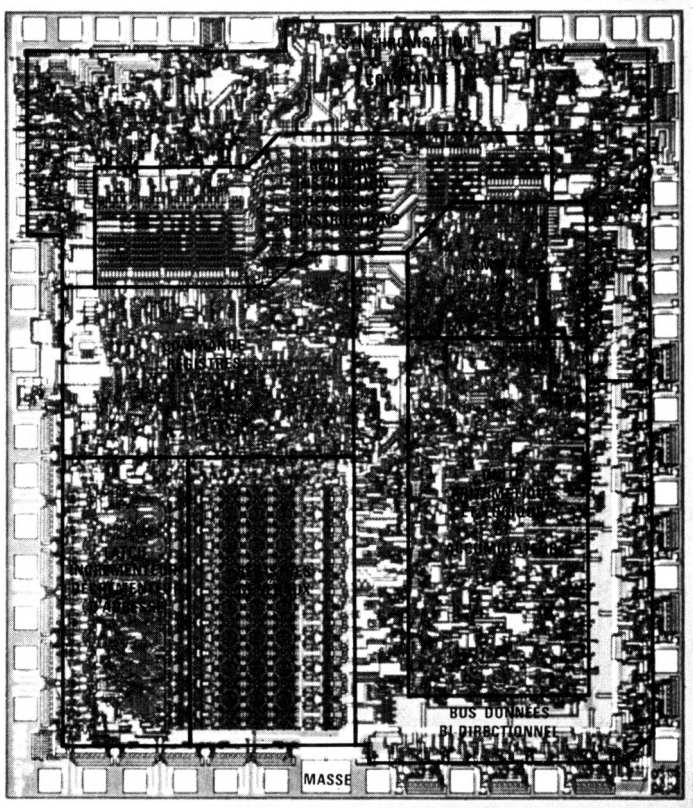
— Les octets suivants (bytes), s'ils existent, seront représentés par $\langle B_2 \rangle$ et $\langle B_3 \rangle$.

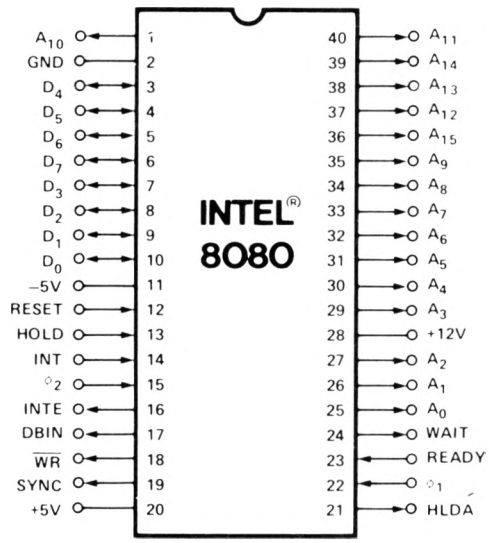
— Nous désignerons par r l'adresse d'un des registres internes A, B, C, D, E, H, L. Ces 8 adresses (en binaire) sont d'ailleurs les suivantes :

B000 soit 0
C001 soit 1
D010 soit 2
E011 soit 3
H100 soit 4
L101 soit 5
M110 soit 6
A111 soit 7

Qu'est-ce que ce **M** vient faire là ?

Si nous disons qu'il symbolise une adresse en mémoire, elle nous paraît plutôt limitée ? Pas d'affolement... Lorsque nous avons décortiqué le 8008, vous vous rappelez du schéma d'adressage assez particulier utilisant H et L comme pointeur, et comme le 8080 a conservé à peu près toutes les instructions du 8008, cette méthode sera très fréquemment utilisée. Rappelons-nous son principe : on charge dans H et L l'adresse mémoire où l'on veut travailler, adresse qui fait donc deux octets de long et





μ photo repérée de la puce du 8080, et correspondance des broches du boitier. On remarque que le bus données passe en profondeur dans le silicium en dessous d'autres éléments pour gagner de la place, d'où augmentation et du nombre de puces fabriquées à la fois sur une tranche et du rendement de production.

permet d'adresser par conséquent 64 Ko. On travaille alors sur la donnée, qui sera en général une chaîne de caractères, à l'aide d'une série d'instructions telles que MOV r, M ou ADD M, où r sera donc un registre et M l'emplacement mémoire dont l'adresse est dans H et L. Rappelons que H (high) est la moitié la plus significative de l'adresse, et L (low) celle de faible poids. On dispose d'instructions permettant d'incrémenter / décrémenter la paire de registres HL avec, comme nous l'avons vu, report automatique de retenue de l'un à l'autre ; pas de piège donc, ils réagissent dans ce cas comme un tout.

— **A** est l'accumulateur, et son contenu figure implicitement comme opérande dans de nombreuses instructions : il a la taille d'un octet.

— **C** (condition) indiquera l'un des bits drapeaux (*flags*) au nombre de quatre :

- **C** *carry* (report), vrai s'il y a un débordement de l'accu ou un passage en dessous de zéro, nous avons déjà vu ses tribulations en arithmétique ; une opération d'addition qui aurait pu produire un report, mais n'en a pas produit, remet C à zéro,
- **Z** *zéro*, vrai si une opération a produit zéro dans un registre, même si elle a provoqué en même temps un report,
- **S** *signe*, vrai si le bit le plus significatif d'un registre est égal à un,
- **P** *parité*, vrai si le nombre de bits 1 d'un octet est pair. Ce test, rarement câblé dans les microprocesseurs, est particulièrement utile.

Il existe un cinquième drapeau, mais nous le mettrons à part car il n'est pas testable par programme. C'est le bit **AC**, *Auxiliary Carry* (report auxiliaire) : il n'est utilisé que par l'instruction d'arithmétique décimale DAA qui permet de réaliser l'addition de deux chiffres décimaux à la fois (quatre bits chacun) dans l'accu. Ce bit indique qu'une retenue s'est propagée à partir du bit 3. Par exemple, l'addition suivante va mettre à zéro le bit C et à un le bit AC :

```
N° d'ordre des bits :   7 6 5 4 3 2 1 0
   2 E H =              0 0 1 0 1 1 1 0
+  7 4 H =              0 1 1 1 0 1 0 0

   A 2 H =              1 0 1 0 0 0 1 0
                        C = 0          AC = 1
```

Le bit AC est affecté par les instructions d'addition, soustraction, incrémentation, décrémentation et comparaison.

Le reste du symbolisme concernant les registres est classique : r_m indique le bit m du registre r (m = 0 pour le bit de droite), SSS signifie *source*, DDD *destination* (adresses de trois bits comme vu) et () le contenu d'un registre. Si certains bits sont remplacés par des X, c'est que leur valeur peut être 1 ou 0. Enfin PC et SP interviendront souvent, mais ce sont de vieilles connaissances.

3 - 8- 4. Les instructions de la machine

Attaquons maintenant bravement les instructions : il y en a beaucoup, mais abondance de biens ne nuit pas ! Nous donnerons pour chacune d'elles le code machine, le code symbolique assembleur et la description de son action avec des exemples.

Instructions du bit de report

Il y en a deux, d'un octet. Code machine :
0011X111

CMC *Complement carry* (complémenter le report), pour X = 1. CMC complémente le bit C (le met à 0 s'il était à 1, à 1 s'il était à 0). Seul le drapeau C est affecté, de même pour
3F

STC *Set carry* (mettre à 1 le report), pour X = 0.
37

Instructions sur un registre

Un registre, ou un emplacement mémoire. Dans ce dernier cas, son adresse est indiquée par le contenu de H & L.

INR *Increment Register or Memory* (incrémenter registre ou mémoire)
00XXX1000 XXX = l'une des 8 adresses vues.

L'octet désigné est augmenté d'une unité. Les bits de condition affectés sont Z, S, P et AC. Par exemple, si le registre C contient 99H, l'instruction INR C (mode d'écriture en assembleur, en langage machine hexa c'est 0C, la présence du C n'étant que coïncidence !) provoquera l'apparition de 9AH dans C.

DCR *Decrement Register or Memory* (décrémenter registre ou mémoire)
00XXX101 XXX = l'une des 8 adresses vues.

L'octet désigné est diminué d'une unité. Les bits de condition affectés sont Z, S, P et AC. Exemple : le registre H contient 3AH, et le registre L contient 7CH. A l'emplacement mémoire d'adresse 3A7CH il y a 40H. Après exécution de DCR M (code machine hexa 35) on y trouvera **3F**.

CMA *Complement Accumulator* (complémenter l'accumulateur)
00101111, soit **2F**.

Chaque bit de l'accumulateur est complémenté : on obtient ainsi le complément à un de son contenu. Aucun bit de condition n'est affecté. Exemple : l'accu contient 51H avant, il contiendra AEH après (01010001 puis 10101110).

DAA *Decimal Adjust Accumulator* (ajustage decimal de l'accumulateur)
00100111, soit **27**.

Voici donc la fameuse instruction qui va nous permettre de pratiquer l'arithmétique décimale sans peine. On l'effectue juste après l'instruction d'addition binaire, et elle permet de remettre en forme correctement en DCB son résultat : c'est la seule instruction dont le déroulement soit affecté par la valeur du bit AC. Le precessus, imaginé dès les premiers ordinateurs et les calculatrices mécanographiques, s'effectue automatiquement en deux temps :

1) Si les quatre bits de droite de l'accu représentent un nombre supérieur à 9, ou si le bit AC vaut 1, l'accumulateur est incrémenté de six. Sinon, on passe sans rien faire à 2).
2) Si les quatre bits de gauche de l'accumulateur représentent maintenant un nombre supérieur à 9, ou si le bit C vaut 1, ces quatre bits sont incrémentés de six. Sinon, rien ne se passe.

Si un report provenant des quatre bits de gauche se produit durant 1), le bit AC est mis à 1 ; sinon, il est remis à 0. De la même façon, si une retenue en provenance de ces mêmes bits se produit durant 2), le bit C est mis à 1 ; sinon, il n'est pas modifié. Dans ce déroulement, les cinq flags sont affectés.
Exemple : il y a 9BH dans l'accu, et les deux bits de report sont à zéro.

1) Comme les bits 0 et 3 représentent un nombre supérieur à 9, 6 y est ajouté. Un report va alors s'échapper de 3 vers 4 et mettre AC à 1 :

```
N° d'ordre des bits :  7654      3210
Accu :                 1001      1011  =  9 B H
+6                               0110
                       1010  R   0001  =  A 1 H
                         AC = 1
```

2) Comme maintenant les bits 4 à 7 représentent un nombre supérieur à 9, 6 y est ajouté. Un report va alors s'échapper de 7 et mettre C à 1 :

```
N° d'ordre des bits :  7654      3210
Accu :                 1010      0001  =  A 1 H
+6                     0110
                     R 0000      0001
                     C = 1
```

Bien sûr, le positionnement de C nous permettra de propager les reports tout au long d'une chaîne de n octets, représentant un nombre décimal de 2n chiffres : nous verrons l'écriture de cette boucle d'addition comme exemple de programmation.

Instruction de non-opération

Il y en a une seule, et elle occupe un octet :
NOP *No Operation* (non-opération)
00000000, soit **00H**

Rien ne se passe (sauf que (PC) est augmenté de 1). On va à l'instruction suivante.

On remarquera que si la mémoire est remise à zéro, elle est remplie de NOP : si à la suite d'une erreur de branchement on atterrit n'importe où, on va filer à toute vitesse vers les adresses les plus élevées.

Instructions de mouvement de données

Elles transfèrent les données octet par octet (mot par mot, l'octet étant le mot du 8080) entre registres ou entre mémoire et registres. Trois instructions d'un octet, MOV, STAX et LDAX, compte tenu des adresses possibles correspondent à 63, 2 et 2 instructions effectives. Aucun flag n'est affecté par ces instructions.

MOVE *Move* (mouvoir)
01DDDSSS L'instruction transfère un octet de la source dans la destination (la source reste inchangée). Compte tenu des huit adresses source et des huit adresses destination possibles, il y a apparemment 64 combinaisons telles que MOV B,A MOV L,E MOV C,M etc. La destination (réception) est indiquée en premier (ici B, L et C) et la source (émission) en second A, E, M). Comme toujours, M est l'emplacement mémoire dont l'adresse est dans H & L. Si l'on a le droit d'écrire MOV A,A ou MOV D,D dont l'effet est évidemment nul (ce sont des NOP déguisés !), on n'a pas le droit d'écrire MOV M,M qui correspondrait (vérifiez-le) au code opération hexa 76 : ce code est en effet celui de l'instruction halte HLT, dont nous avons vu l'usage bien particulier.

STAX et **LDAX** sont des instructions de mouvement permettant de décharger et charger l'accumulateur, les emplacements mémoire correspondants étant indiqués par le contenu de la paire de registre B (B & C) ou D (D & E). Le code machine général est :

000XX010

premier	X : 0 = B,	1 = D
deuxième	X : 0 = STAX,	1 = LDAX

STAX *Store Accumulator* (stocker l'accu à l'adresse indiquée par...)
Le contenu de l'accu est stocké dans le mot mémoire dont l'adresse est dans B ou D. Deux

écritures donc (recalculez comme exercice les codes machine) :

STAX B	code machine	**02**
STAX D	code machine	**12**

LDAX *Load Accumulator* (charger l'accu à partir de l'adresse indiquée par...)

Le contenu du mot mémoire dont l'adresse est dans B ou D va dans A :

LDAX B	code machine	**0A**
LDAX D	code machine	**1A**

Instructions arithmétiques et logiques de l'accumulateur utilisant registres ou mémoire

Ces huit instructions d'un octet ont toutes un code machine débutant par 10. Les trois bits suivants YYY indiquent la nature de l'instruction, et les trois derniers AAA l'une des huit adresses possibles, toujours les mêmes : les six registres, l'accu lui-même, et la mémoire avec adresse dans H & L.

Ces adresses revenant sans arrêt, on peut tenter quelques combinaisons mnémoniques simples pour en retrouver la liste. Comme elle est dans l'ordre alphabétique sauf pour A, il suffit de se rappeler d'un ou deux pour recaler le tout : ce bêta est un zéro, c'est un, Deux ou D², l'étroit mousquetaire (film bien connu), H IV (le lycée...), RAM6 (à faire frémir tout égyptologue), l'accu est chargé (111)... En vous forçant un peu, vous trouverez peut-être mieux ! La structure de l'octet est donc :

10YYYAAA YYY étant égal à :

000 pour ADD	100 pour ANA
001 pour ADC	101 pour XRA
010 pour SUB	110 pour ORA
011 pour SBB	111 pour CMP

Le contenu de l'octet adressé reste inchangé, le résultat de l'instruction remplace le contenu précédent de l'accu.

ADD *Add register or memory to accumulator* (ajouter registre ou mémoire à l'accumulateur)

10000 AAA, soit **8** suivi de **0** à **7**.

Le contenu de l'octet spécifié est ajouté au contenu de l'accu en utilisant l'arithmétique en complément à deux que nous avons étudiée. Les cinq drapeaux de condition sont affectés.

Exemple 1 : le registre D contient 2EH et l'accu 6CH. L'instruction ADD D (soit 82, est-ce que les mnémoniques marchent ?) va donner :

2EH = 00101110
6CH = 01101100

10011010 = 9AH

Les bits Z et C sont remis à zéro ; les bits P et S sont mis à un, de même que le bit AC puisqu'il y a eu un report issu du bit 3. Le contenu de l'accu est devenu 9AH.

Exemple 2 : l'instruction ADD A (87) double le contenu de l'accu, ce qui revient à décaler son contenu d'un cran à gauche, un zéro entrant à droite (de même que multiplier par dix en décimal décale d'un cran à gauche un nombre décimal).

ADC *Add register or memory to accumulator with carry* (ajouter registre ou mémoire à l'accumulateur avec report)

10001AAA, soit **8** suivi de **8** à **F**

L'octet spécifié plus le contenu du bit C sont ajoutés au contenu de l'accu. Les cinq flags sont affectés.

Exemple 1 : le registre C contient 3DH, l'accu 42H et le bit C = 0. L'instruction ADC C (89) effectuera :

3DH = 00111101
42H = 01000010
C = 0

01111111 = 7FH qui apparaîtra dans A

A l'issue de l'opération, les cinq flags seront tous à zéro.

Exemple 2 : identique, mais cette fois C = 1 :

3DH = 00111101
42H = 01000010
C = 1

10000000 = 80H dans A

A l'issue de l'opération C = 0, S = 1, Z = 0, P = 0 et AC = 1.

Notez bien toutes ces actions sur les drapeaux, elles servent constamment en programmation. L'utilité évidente de cette instruction est l'addition de deux chaînes d'octets avec propagation de report tout du long.

SUB *Subtract register or memory from accumulator* (soustraire registre ou mémoire de l'accumulateur)

10010AAA, soit **9** suivi de **0** à **7**

L'octet spécifié est soustrait de (A) en utilisant l'arithmétique en complément à deux.

Rappelons que s'il n'y a *pas* de report s'échappant de la position de gauche au moment de l'addition du complément, ce qui indique qu'une retenue soustractive s'est produite, le bit Carry est *mis à 1* ; sinon, il est mis à 0 (c'est le contraire de l'addition qui met C à 0 s'il n'y a pas de report).

Tous les flags sont affectés.

Exemple : l'accu contient 3EH. L'instruction SUBA

(97) va soustraire l'accu de lui-même, ce qui permet donc de le remettre à zéro, *ainsi que le bit C*, en une seule instruction :

```
      3EH = 00111110
+(—3EH) = 11000001   complément de chaque
                1    bit et + 1 pour
                     complément à 2.
Report 1 00000000 = 0
```

Comme il y eu report et que nous sommes en opération de soustraction, le μP met C à zéro : C = 0.

Il y a eu un report interne de A_3 : donc AC = 1.

Le nombre de bits A est nul, donc pair, donc P = 1.

Le résultat de l'opération est zéro, donc Z = 1 pour le signaler.

Enfin, pas de 1 à gauche, donc pas de signe moins, donc S = 0.

SBB *Subtract register or memory from accumulator with borrow* (soustraire registre ou mémoire de l'accumulateur est retenue)

10011AAA, soit **9** suivi de **8** à **F**

Le bit C est ajouté au contenu de l'octet spécifié, puis le résultat ainsi obtenu est soustrait de (A) en arithmétique en complément à 2. Cette instruction est donc celle que nous utiliserons pour la soustraction l'une de l'autre des deux chaînes d'octets, avec propagation des retenues soustractives. Les cinq bits de condition sont affectés.

Exemple : le registre r contient 2, l'accu 4 et C = 1. On aura :

02H + C = 03H, dont le complément à 2 est 11111101.

Il est alors ajouté à (A) pour réaliser la soustraction :

```
04H = 00000100
      11111101

Report 1 00000001 = 01H
```

Le report qui s'échappe remet C à 0 puisque nous sommes en soustraction, et le résultat final dans l'accu est 1. On aura donc Z = 0, AC = 1 (report de A_3), P = 0 et S = 0.

ANA *Logical AND register or memory with accumulator* (ET logique entre accumulateur et registre ou mémoire)

10100AAA, soit **A** suivi de **0** à **7**

Voici donc notre première instruction logique ! Elle réalise l'intersection logique de chaque bit de A avec le bit de même rang dans l'octet spécifié, le résultat se trouvant dans A. De plus, C est mis à 0. Rappelons que le résultat du ET est zéro, sauf dans le cas où les bits valent 1 tous les deux. Quatre flags affectés : C, Z, S et P.

Exemple : l'une des utilisations les plus importantes de cette instruction est pour les masques d'extraction. Elle permet en effet de mettre sélectivement à zéro des bits dans un octet, laissant intouchés les groupes de bits (ou bits isolés) intéressants.

Supposons que nous ayions chargé un octet dans l'accu, et que nous voulions mettre à zéro les quatre bits de gauche sans toucher au reste ; nous chargerons par exemple dans B la valeur 0FH (ce sera notre masque) et effectuerons ensuite ANA B (A0). S'il se trouve que l'accu contient par exemple B5H, nous aurons :

```
Accu = 10110101 = B5H
reg. C = 00001111 = 0FH

intersection 00000101 = 05H dans l'accu
```

et le tour est joué. Il nous sera alors loisible de réaliser nos traitements, tests, etc. sur la partie sélectionnée.

XRA *Logical Exclusive-Or register or memory with accumulator (zero accumulator)* (OU logique exclusif entre accumulateur et registre ou mémoire (RAZ accumulateur))

10101AAA, soit **A** suivi de **8** à **F**

Un OU exclusif est réalisé bit à bit entre l'octet désigné et (A), le résultat se trouvant dans A. Le bit C est mis à 0.

Rappelons que le OU exclusif, OUX si vous préférez, donne 1 si les valeurs des deux bits sont différentes. Les flags C, Z, S et P sont affectés.

Exemple 1 : cette instruction permet de remettre à zéro l'accu. Si nous voulons faire RAZ de B et E par exemple, nous écrirons :

assembleur	absolu
XRA A	AF
MOV B,A	47
MOV E,A	5F

Exemple 2 : un OUX entre 1 et un bit quelconque complémente ce dernier. En effet, 0 OUX 1 = 1 et 1 OUX 1 = 0. Si donc A contient uniquement des 1 (soit (A) = FF), l'instruction XRA B (A8) donnera le complément à 1 de (B) dans l'accu.

Exemple 3 : test d'un changement d'état. On utilise souvent en programmation les bits d'un octet que l'on positionne individuellement à 1 ou 0, au fil d'un programme, comme indicateurs d'états variés de processus, de variables, de comparaisons, etc. On dispose ainsi de huit drapeaux dans un mot d'état, et on aura par exemple besoin de savoir lesquels ont changé depuis la dernière fois où nous avons stocké ledit mot d'état à l'adresse symbolique ETAT1 par exemple. Le nouveau mot sera supposé dans l'octet adjacent, un cran plus près, en ETAT2. Préalablement, l'adresse ETAT2 a été mise dans HL :

```
assembleur                                  absolu
MOV A,M  ; (ETAT2) dans A                   7E
INX H    ; progression adresse              23
MOV B,M  ; (ETAT1) dans B                   46
XRA B    ; OUX des états                    A8
ANA B    ; ET du résultat avec (ETAT1)      A0
```

Remarquons d'abord qu'en assembleur nous séparons les commentaires des instructions par un point-virgule. Pour la clarté nous avons mis en majuscules les noms symboliques ou opérateurs, et en minuscules le reste. En réalité, sur les listings tout est en majuscules.

Notre instruction XRA va mettre un bit 1 dans l'accu à l'emplacement de chaque bit d'état ayant changé de valeur. Prenons un exemple précis :

```
N° d'ordre des bits          7 6 5 4 3 2 1 0
(ETAT1) = 5 CH =             0 1 0 1 1 1 0 0
(ETAT2) = 78 H =             0 1 1 1 1 0 0 0

OUX donne                    0 0 1 0 0 1 0 0
```

dans l'accu, l'état des bits 2 et 5 ayant changé. Nous voulons maintenant savoir dans quel sens s'est produite la variation. C'est ce que va nous donner le ET logique avec (ETAT1) :

```
Résultat = 00100100
(ETAT 1) = 01011100

ET donne 00000100
```

Le bit 2 étant à 1, ceci indique qu'il a été mis à un entre l'échantillonnage de ETAT1 et de ETAT2. Le bit 5 étant à 0, ceci indique qu'il a, lui, été mis à 0.

ORA *Logical OR register or memory with accumulator* (OU logique entre accumulateur et registre ou mémoire)
10110AAA, soit **B** suivi de **0** à **7**

Un OU inclusif est réalisé bit à bit, entre l'octet désigné et (A). Le bit C est mis à 0.

Rappelons que le OU de deux bits n'est égal à 0 que si les deux bits sont eux-mêmes nuls. Les drapeaux C, Z, S et P sont affectés.

Exemple : cette instruction est souvent utilisée pour forcer à un des bits ou groupes de bits, par exemple des drapeaux dans un mot d'état (pour les forcer à zéro, on utilise ANA). Supposons que le registre C contienne 0FH, et l'accu 33H. L'instruction 0RA C (code hexa B 1) donnera :

```
Accu      = 00110011 = 33H
Registre C = 00001111 = 0FH
Résultat  = 00111111 = 3FH dans l'accu
```

Les quatre bits de gauche sont inchangés, et les quatre bits de droite sont forcés à 1.

CMP *Compare register or memory with accumulator* (comparer accumulateur à registre ou mémoire)
10111AAA, soit **B** suivi de **8** à **F**

L'octet spécifié est comparé au contenu de l'accu. Cette comparaison s'effectue par soustraction interne de l'octet à l'accu (l'un et l'autre restant inchangés) et positionnement des drapeaux en fonction du résultat. En particulier on aura Z = 1 si les quantités sont égales, et Z = 0 si elles sont inégales. Le bit C sera mis à 1 si le contenu de l'octet est supérieur à celui de l'accu, car dans ce cas la soustraction n'aura pas produit de report à gauche. On aura C = 0 dans le cas contraire. Les cinq drapeaux sont affectés.

Attention ! Si les deux quantités à comparer sont de signe différent, l'état du bit C sera inversé... c'est une chausse-trappe à bien surveiller.

Exemple 1 : on a 0AH dans A et 05H dans E. L'instruction CMP E (BB) va donner :

```
          Accu = 0AH = 00001010
+(—Registre E) =  5H = 11111011

          Report 1   00000101 = Résultat
```

Le report remet C à 0. Après l'opération, il y a toujours 0AH dans A et 05H dans E. Le fait que C = 0 et Z = 0 indique que (E) est inférieur à (A).

Exemple 2 : cette fois, l'accu contient 02H :

```
          Accu = 02H = 00000010
+(—Registre E) =  5H = 11111011

          Pas de report   11111101 = Résultat
```

On a Z = 0 et C = 1, indiquant (E) supérieur à (A).

Exemple 3 : on a —1BH dans l'accu :

```
          Accu = —1BH = 11100101
+(—Registre E) = — 5H = 11111011

          Report 1   11100000 = Résultat
```

En soustraction, le report donne C = 0. Les signes de (A) et (E) étant différents, cette condition indique (E) supérieur à (A).

Instructions de rotation de l'accumulateur

Ce sont des instructions d'un octet dont le code est :

```
000XX111   avec comme valeur de XX :
           00 pour RLC          10 pour RAL
           01 pour RRC          11 pour RAR
```

RLC *Rotate accumulator left* (rotation à gauche de l'accumulateur)
00000111 soit **07**

Le bit C est mis à la vleur du bit de gauche de l'accu. Le contenu de ce dernier est décalé d'un cran à gauche, le bit sortant à gauche étant réinjecté à droite. Le seul drapeau affecté est C.
Exemple : il y a F2H dans l'accu :

avant RLC : C = X Accu = 11110010
après RLC : C = 1 Accu = 11100101

RRC *Rotate accumulator right* (rotation à droite de l'accumulateur)
00001111 soit **0F**

Le bit C est mis à la valeur du bit de droite de l'accu. Le contenu de ce dernier est décalé d'une position à droite, le bit sortant à droite étant réinjecté à gauche. Le seul drapeau affecté est C.
Exemple : il y a toujours F2H dans l'accu :

avant RRC : C = X Accu = 11110010
après RRC : C = 0 Accu = 01111001

RAL *Rotate accumulator left through carry* (rotation à gauche de l'accumulateur à travers le report)
00010111 soit **17**

Le contenu de l'accu est décalé d'un cran à gauche. Le bit sortant rentre dans C, et le précédent contenu de C rentre à droite dans l'accu. Seul le drapeau C est affecté.
Exemple : il y a B5H dans A :

avant RAL : C = 0 Accu = 10110101
après RAL : C = 1 Accu = 01101010

RAR *Rotate accumulator right through carry* (rotation à droite de l'accumulateur à travers le report)
00011111 soit **1F**

Le contenu de l'accu est décalé d'un cran à droite. Le bit sortant rentre dans C, et le précédent contenu de C rentre à gauche dans l'accu. Seul le drapeau C est affecté.
Exemple : il y a 6AH dans A :

avant RAR : Accu = 01101010 C = 1
après RAR : Accu = 10110101 C = 0

Instructions sur paires de registres

PUSH *Push data onto stack* (entasser les données sur la pile)
11AA0101, soit **C5, D5, E5, F5**

Car AA représente les adresses de paires de registres ou du PSW comme suit :

00 pour B et C 10 pour H et L
01 pour D et E 11 pour le PSW

(rappelons que le PSW comprend (A) et les cinq drapeaux, et remarquons que SP ne figure pas dans la liste : on l'atteint autrement).

Cette instruction stocke le contenu de la paire de registre spécifiée dans les deux octets mémoire indiqués par le pointeur de pile SP. Nous avons déjà vu le fonctionnement de SP dans la description hardware détaillée de CALL : *la pile se développe vers les adresses faibles*. Quand on exécute PUSH, la valeur courante de SP est d'abord décrémentée de 1, et c'est à cette adresse mémoire SP—1 que le contenu du registre de plus fort poids (B, D, H ou l'Accu, de fort poids puisqu'il est en plomb) est stocké ; puis nouvelle décrémentation, et en SP—2 sera stocké le registre de faible poids (C, E, L ou les 5 drapeaux, légers). Aucun drapeau n'est affecté. Rappelons-nous que les registres de poids fort sont stockés dans l'adresse forte, et ceux de poids faible dans l'adresse faible.

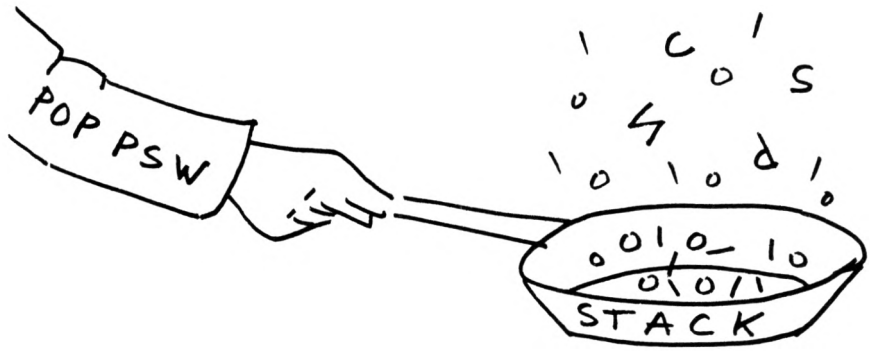
La configuration de l'octet des drapeaux est la suivante :

S,Z,0,AC,0,P,1,C
N° d'ordre : 7 6 5 4 3 2 1 0

Cette instruction est très pratique pour évacuer rapidement deux octets d'un seul coup du μP vers la mémoire, sans même avoir à charger préalablement l'adresse de leur destination dans HL. Et naturellement, elle permet de sauvegarder l'état du μP en stockant le PSW, avant d'effectuer le service d'une interruption par exemple.
Exemple 1 : on a 8FH dans D, 9DH dans E et le SP contient 3A2CH. On effectue PUSH D (D5), et l'on trouve ensuite : 8F dans l'octet d'adresse 3A2B, 9D en 3A2A, toujours 8F9D dans le registre de 16 bits DE, et le registre SP contient 3A2A (après être passé par 3A2B).
Exemple 2 : on a 1FH dans A ; le SP est à 502AH ; C, Z et P valent 1 ; S et AC valent 0. On exécute PUSH PSW (F5) et l'on trouve ensuite : 47 en 5028, 1F en 5029 (vérifiez ce 47 sur un bout de papier) et (SP) = 5028.

POP *Pop data off stack* (récupérer les données de la pile)
11AA0001 soit **C1, D1, E1, F1** AA : comme dans PUSH.

La saveur du *pop* est difficile à traduire : faire sauter les bits... comme le pop corn dans la pile/poêle... Bien sûr, cette instruction agit en sens inverse de la précédente : l'octet vers lequel pointe (SP) est chargé en faible poids, puis celui en (SP) + 1 (eh oui, on revient dans l'autre sens) dans les poids forts. Puis dernière incrémentation, et c'est l'adresse (SP) + 2 qui reste dans SP, tout prêt à redémarrer. Aucun drapeau n'est affecté en principe, *mais* si l'instruction exécutée est POP PSW, les cinq drapeaux courants vont être remplacés par ceux provenant de la pile.



Exemple 1 : on a 3D et 93 dans les octets d'adresses 1239 et 123A, et le SP contient 1239. On exécute POP H (E1), et l'on trouve ensuite : 3D dans L, 93 dans H et 123B dans SP.
Exemple 2 : on a C3 en 2C00 et FF en 2C01, et (SP) = 2C00. On exécute POP PSW (code F1), et l'on va trouver : FF dans A, et S = 1, Z = 1, AC = 0, P = 1 et C = 1 (vérifiez encore la configuration, c'est un bon exercice).

DAD *Double add* (double addition)
00AA1001 soit **09, 19, 29, 39**
Cette fois, une variante pour les adresses AA : *il n'y a plus PSW, il est remplacé par SP*. On a donc :

00 pour BC	10 pour HL
01 pour DE	11 pour SP

Le nombre de 16 bits de la paire de registres spécifiée est ajouté au nombre de 16 bits contenu dans HL, en arithmétique en complément à deux. Le résultat remplace l'ancien contenu de HL. Seul le flag C est affecté.
Exemple 1 : on a 33 dans B, 9F dans C, A1 dans H et 7B dans L. On exécute DAD B (09) : résultat, D51A dans HL. Comme il n'y a pas eu de report, C est mis à 0.
Exemple 2 : DAD H (code 29) va doubler le contenu de HL, ce qui est équivalent à décaler l'ensemble des 16 bits d'une position à gauche (un bit zéro entre à droite).

INX *Increment register pair* (incrémenter paire de registres)
00AA0011 soit **03. 13, 23, 33**
Mêmes adresses AA que pour DAD.
Le nombre de 16 bits contenu dans le double registre spécifié est augmenté de 1. Aucun bit de condition n'est affecté.
Exemples : on a 38FF dans DE. L'instruction INX D(13) nous y donnera 3900. Si l'on a FFFF dans SP, INX SP (33) nous y donnera 0000.

DCX *Decrement register pair* (décrémenter paire de registres)
00AA1011 soit **0B, 1B, 2B, 3B**
Mêmes adresses AA que pour DAD.
Le nombre de 16 bits contenu dans la paire de registres spécifiée est diminué de 1. Aucun bit de condition n'est affecté.
Exemple : si HL contient 9800, DCX H (code 2B) y donnera 97FF.

XCHG *Exchange registers* (échanger registres)
11101011 soit **EB**

Pas de variantes cette fois. XCHG échange les contenus de HL et DE. Aucun flag affecté.

Exemple : on a 00 dans H, FF dans L, 33 dans D et 55 dans E. Après exécution de XCHG, on trouvera 33 dans H, 55 dans L, 00 dans D et FF dans E.

XTLH *Exchange stack* (échanger pile)
11100011 soit **E3**

Cette instruction échange le contenu de L avec celui de l'octet en mémoire dont l'adresse est dans SP. Puis le contenu de H est échangé avec celui de l'octet dont l'adresse vaut une unité de plus que le contenu de SP. Aucun flag n'est affecté. SP reste inchangé.

Rappelons-nous au passage que nous avons cité cette instruction dans un paragraphe précédent comme étant la plus complexe que peut exécuter les 8080, puisqu'elle exige le maximum possible d'états successifs, soit 18.

Exemple : il y a 10AD dans SP, 0B3C dans HL, F0 dans le mot mémoire d'adresse 10AD et 0D dans celui d'adresse 10AE. Exécutons XTHL, il vient : 0DF0 dans HL, 3C dans 10AD et 0B dans 10AE ; (SP) reste inchangé et égal à 10AD.

SPHL *Load SP from H and L* (changer H et L dans SP
11111001 soit **F9**

La donnée de 16 bits contenue dans HL remplace le contenu de SP. Le contenu de HL reste inchangé, et aucun drapeau n'est affecté.
Exemple : on a 50 dans H et 6C dans L. Après SPHL, le SP contient 506C.

Instructions à opérande immédiat

Ce sont des instructions bien pratiques — et n'oublions pas au passage qu'elles n'existent par sur des mini-ordinateurs tels que le PDP-8 ! profitons de cette occasion pour rappeler que vis à vis des minis à mots courts, et même du 360, le 8080, machine à octets à longueur variable, présente des avantages déterminants en programmation :

— adresses entières de 16 bits, non tronquées, supprimant les pénibles contraintes de la pagination : ici, plus de découpage de la mémoire en lopins séparés par des barrières qui se dressent en travers du processus normal de la programmation au milieu des boucles et des branchements, forçant à une attention constante et à des changements incessants de contenu des registres de base. La mémoire est un champ homogène qu'on peut adresser partout sans surprise dans une étendue de 65 536 octets.

— il y a des registres généraux en plus de l'accumulateur, ce qui évite un grand nombre d'opérations de chargement et de déchargement, et par ailleurs accélère le déroulement des programmes par la rapidité d'exécution des instructions de registre à registre.

— nous disposons d'instructions immédiates. Rappelons leur fonction : elles utilisent comme opérande le contenu d'une partie de l'instruction elle-même, ce qui permet d'éviter de nombreuses constantes, rend les programmes plus compacts, plus rapides d'exécution et surtout plus faciles à écrire et lisibles.

Ces dernières instructions ont deux ou trois octets de long.

Le premier octet comporte le code opération permettant la reconnaissance de l'instruction, et l'adresse du registre utilisé, en deux bits si c'est un registre double (mêmes adresses que pour DAD) ou en trois bits si c'est un registre simple (de bêta-zéro à Ramsès et l'accu chargé...). Pour les huit instructions spécialisées de l'accumulateur, son adresse est par contre implicite.

Le deuxième octet contient l'opérande immédiat. Dans le cas de LXI, il y a un troisième octet, l'opérande immédiat ayant 16 bits de long (c'est la seule instruction de ce type).

LXI *Load register pair immediate* (charger opérande immédiat dans paire de registres)
00AA0001, soit **01, 11, 21, 31,** suivi de deux octets.

Le deuxième octet de l'instruction est chargé dans le registre de *droite* de la paire (le moins significatif), et le troisième dans l'autre. Attention à cette inversion, signalée dans «Format des instructions et des données», qui est un piège au début quand on programme en hexa au pupitre d'Alcyane : pour les instructions de trois octets, après l'octet de code opération on entre la moitié droite de l'adresse, puis la gauche.

En assembleur, cette inversion est réalisée par le programme d'assemblage. Ceci nous amène à rappeler la méthode la plus facile pour programmer en langage machine :

1. Commencer par écrire le programme en assembleur.
2. Remplacer les adresses symboliques par les adresses réelles en hexa, en réalisant l'inversion signalée.
3. Remplacer les codes opérations mnémoniques par les codes en hexa.
4. Entrer le programme en machine octet par octet seul ou à deux (on va beaucoup plus vite à deux, l'un épelant les octets et vérifiant l'affichage réalisé par l'autre). Pour cette dernière opération, il suffit d'afficher une seule fois l'adresse du début du programme, car ensuite la progression d'adresse est réalisée automatiquement par l'électronique de la face avant d'Alcyane.

Revenons à notre instruction LXI : bien qu'on ne retrouve pas le X dans le libellé de l'instruction, comme nous l'avons vu pour STAX par exemple, il signifie ici adresse (le rôle le plus fréquent des paires de registres est en effet l'adressage indirect). Aucun drapeau n'est affecté par LXI.

Exemple 1 : supposons que le label DEPT désigne l'emplacement mémoire d'adresse 103 H (= 259D). Les trois instructions suivantes produiront le même effet (charger 01H dans H et 03H dans L) :

```
LXI H, 103H
LXI H, 259
LXI H, DEPT
```

On voit donc le choix existant en assembleur pour exprimer cette instruction qui, lorsqu'elle sera transformée en langage machine, deviendra de toutes façons 21, 03, 01 en hexa, en trois octets.

Exemple 2 : l'instruction suivante charge EBC3H dans SP :

```
LXI SP, EBC3H
```

En absolu elle s'écrira 31, C3, EB.

MVI *Move Immediate Data* (mouvoir donnée immédiate)
00AAA110 soit **06, 0E, 16, 1E, 26, 2E, 36, 3E**
suivi d'un octet.

Le contenu du deuxième octet de l'instruction est stocké à l'emplacement indiqué par AAA (registre ou mémoire). Aucun flag n'est affecté. Rap-

pelons que l'adresse mémoire (s'il y a lieu) est contenue dans HL.
Exemple : écrivons la séquence suivante d'instructions :

assembleur	absolu
MVI H,ECH	26EC
MVI L,F4H	2EF4
MVI M,FFH	36FF

La première instruction met 3C dans H, la deuxième F4 dans L et la troisième envoie FF à l'emplacement mémoire d'adresse 3CF4. On ferait mieux de remplacer les deux premières instructions par LXI H, 3CF4H soit 21, F4, EC.

ADI *Add Immediate to Accumulator* (addition immédiate à l'accumulateur)
11000110 soit **C6** suivi de l'octet contenant la donnée.
Le contenu du deuxième octet de l'instruction est additionné au contenu de l'accumulateur en arithémtique en complément à deux. Les cinq drapeaux sont affectés.
Exemple : soit la séquence de trois instructions :

	assembleur		absolu	
label	code	opérande	adresses	instructions
TAI :	MVI	A,20	0A19	3E14
TA2 :	ADI	66	0A1B	C642
TA3 :	ADI	-66	0A1D	C6BE

Plusieurs remarques : les labels (adresses des instructions) en assembleur doivent être suivis de deux points, et l'assembleur traduit comme déjà vu le décimal en hexa.
L'instruction logée en mémoire et débutant à l'octet d'adresse symbolique TA1 (que nous avons supposée être 0A19H) va charger 14H dans l'accu. L'instruction en TA2 va accomplir l'addition suivante :

Accumulateur : 14H = 00010100
Opérande immédiat : 42H = 01000010

Nouveau contenu accu : 01010110 = 56H

Le bit de parité est mis à un, et tous les autres flags sont mis à zéro (pas de report, résultat non nul, etc.).
La troisième instruction va restaurer le contenu initial de l'accumulateur en exécutant l'addition suivante :

Accumulateur : 56H = 01010110
Donnée dans TA3 : BEH = 10111110

Résultat : = 00010100 = 14H

Les bits C, AC et P sont mis à un. Les bits Z et S sont mis à zéro.

Sur la feuille en langage machine, on peut évidemment si on le désire, disposer (par exemple) ainsi le programme :

adresses	informations	assembleur
0A19	3E	MVI A
0A1A	14	20
0A1B	C6	ADI
0A1C	42	66
0A1D	C6	ADI
0A1E	BE	— 66

L'arithmétique utilisée ici n'est pas particulièrement agréable, ce programme travaillant en binaire. Dans la plupart des cas pratiques, nous vous conseillons d'utiliser le décimal, sous la forme de Décimal Codé en Binaire bien entendu, quatre bits par chiffre décimal. Au pupitre d'Alcyane, nous pourrons alors entrer ou sortir les nombres décimaux par deux chiffres à la fois, avec des configurations faciles à lire de zéro à neuf.

ACI *Add Immediate to Accumulator with Carry* (addition immédiate à l'accumulateur avec report)
11001110 soit **CE** suivi de l'octet contenant la donnée.

Le contenu du deuxième octet de l'instruction est additionné comme précédemment à l'accumulateur, et le bit de report C au tout. Les cinq flags sont affectés.
Exemple :

	assembleur		absolu	
label	code	opérande	adresses	instructions
K1 :	MVI	A,56H	FB3D	3E56
K2 :	ACI	-66	FB3F	CEBE
K3 :	ACI	66	FB41	CE42

Si nous supposons que C = 0 juste avant que l'instruction qui est en K2 soit exécutée, cette instruction donnera le même résultat que celle précédemment en TA3, soit 14H dans l'accu et C = 1 (notons à ce propos que le flag C est souvent aussi appelé CY, pour *carry*, dans les brochures d'Intel, et nous emploierons donc tantôt C tantôt CY). Si vous ne trouvez pas la même chose... c'est peut-être que vous n'avez pas remarqué que décimal et hexa ont été méchamment mélangés dans l'exemple de programmation ! Un point pour vous si tout est correct.

L'instruction qui est en FB41 (alias K3) exécute alors l'addition suivante :

Accumulateur : 14H = 00010100
Opérande immédiat : 42H = 01000010
CY = 1 = 00000001

Résultat 01010111 = 57H

SUI *Subtract Immedite from Accumulator* (soustraction immédiate de l'accumulateur)
11010110 soit **D6** suivi de l'octet contenant la donnée.

La donnée immédiate contenue dans le deuxième octet est soustraite (toujours en arithmétique en complément à deux) du contenu de l'accumulateur. Nous effectuons donc une opération de soustraction (la façon dont elle se passe, par complémentation puis addition, nous importe peu, *c'est une soustraction*). Rappelons une fois de plus que dans ce cas CY est mis à zéro si un report s'échappe à gauche (tout ceci donc inverse du cas de l'addition, ce qui est le moyen le plus simple de s'en rappeler). Les cinq flags sont affectés.
Exemple : On peut utiliser cette instruction pour obtenir un effet équivalent à DCR A déjà vu :

	assembleur		absolu	
label	code	opérande	adresse	instruction
	MVI	A,0	2AB9	3E00
S1 :	SUI	1	2ABB	D601

La première instruction met l'accu à zéro, et la deuxième effectue :

Accumulateur = 0H = 00000000
— Donnée immédiate = —1H = 11111111*

Résultat 11111111

* (en complément à deux)

et CY = 1 puisqu'il n'y a pas eu de report. Les bits Z et AC sont mis à zéro, les bits S et P à un.

SBI *Subtract Immediate with Borrow* (soustraction immédiate avec retenue)
11011110 soit **DE** suivi de l'octet-donnée.
Le bit CY est ajouté par le μP à l'octet de donnée immédiate. Le résultat est alors soustrait du contenu de l'accu.
Cette fois donc le report est pris en compte, ce qui est utile pour des opérations sur plusieurs octets. Les cinq drapeaux sont affectés, et en particulier CY (selon la règle rappelée pour SUI).
Exemple :

	assembleur		absolu	
label	code	opérande	adresse	instruction
	XRA	A	C23F	AF
	SBI	1	C240	DE01

L'instruction XRA fait RAZ de l'accu comme déjà vu. Si CY = 0, l'instruction SBI donnera :

Donnée immédiate + CY = 01H
Complément à 2 de 01H = 11111111

Il est alors ajouté à l'accu :

Accu = 00000000
11111111

11111111 = —1H

Comme aucun report ne s'est produit, on aura CY = 1, Z et AC seront à 0 et S et P à 1.
Supposons maintenant qu'au début CY = 1 :

Donnée immédiate + C = 02H
Complément à 2 = 11111110

puis ensuite :

Accu = 00000000
11111110

11111110 = —2H

Toujours pas de report, donc CY = 1 ainsi que S, tandis que Z, P et AC seront nuls.

ANI *And immediate with accumulator* (ET immédiat avec l'accumulateur)
11100110 soit **E6** suivi de l'octet donnée
Une intersection logique (voir l'instruction ANA) est réalisée entre la donnée immédiate et le contenu de l'accu. Le flag CY est mis à zéro, et les flags Z, S et P sont affectés.
Exemple :

	assembleur		absolu	
label	code	opérande	adresse	instruction
	MOV	A,C	578F	79
A1 :	ANI	0FH	5790	E60F

Le contenu du registre C est amené dans l'accu. L'instruction ANI remet alors à zéro les quatre bits de gauche, ceux de droite restant inchangés. Z ne sera mis à un que si ces derniers bits étaient préalablement nuls.
S'il y a, par exemple, 3AH dans C, le déroulement sera le suivant :

Accumulateur = 3AH = 00111010
Opérande immédiat = 0FH = 00001111
Intersection
00001010 = 0AH

XRI *Exclusive Or Immediate with Accumulator* (OUX immédiat avec l'accumulateur)
11101110 soit **EE** suivi de l'octet donnée.

Un OU logique exclusif (voir XRA) est réalisé entre la donnée immédiate et le contenu de l'accu. CY est mis à zéro, et Z, S et P sont affectés.

Exemple : comme tout bit OUXé avec un est complémenté, et que tout bit OUXé avec zéro est inchangé, il est possible par XRI de complémenter certains bits de l'accu. Par exemple, XRI 81H complémentera les bits situés aux deux bouts de l'accu. Si ce dernier contient 3BH, nous aurons :

Accumulateur = 3BH = 00111011
Accumulateur = 3BH = 00111011
Opérande immédiat = 81 H = 10000001

Résultat 10111010 = BAH !

ORI *Or Immediate with Accumulator* (OU immédiat avec l'accumulateur)
11110110 soit **F6** suivi de l'octet donnée

Un OU logique inclusif (voir instruction ORA) est réalisé, entre l'octet donnée et (A). CY est mis à zéro, Z, S et P sont affectés, c'est-à-dire positionnés à un ou zéro selon les caractéristiques du résultat.

Exemple :

	assembleur		absolu	
label	code	opérande	adresse	instruction
	MOV	A,C	7777	79
L1 :	0RI	0FH	7778	760F

La première instruction amène (C) dans A, et la deuxième met à un la moitié droite de l'accu, la moitié gauche étant inchangée. Si (C) vaut B5H, on a :

Accumulateur = B5H = 10110101
Opérande immédiat = 0FH = 0000FFFF

Résultat 10111111 = BFH

CPI *Compare Immediate with Accumulator* (comparaison immédiate avec l'accumulateur)
11111110 soit **FE** suivi de l'octet donnée

L'opérande immédiat d'un octet est comparé au contenu de l'accu.

Cette comparaison s'effectue comme décrit en détail dans CMP, les cinq flags étant affectés, et CY et Z indiquant le résultat de la comparaison.

Exemple :

	assembleur		absolu	
label	code	opérande	adresse	instruction
	MVI	A,4AH	3713	3E4A
	CPI	40H	37A5	FE40

Ceci donnera :

Accumulateur = 4AH = 01001010
+ (—opérande immédiat) = —40H = 11000000

Résultat Report 1 00001010

Le report, comme nous sommes en soustraction, met CY à zéro. L'accu contient toujours 4AH, mais le bit Z sera mis à zéro, indiquant que les quantités sont inégales. CY sera mis à zéro, indiquant que la donnée immédiate est inférieure à l'accumulateur (cas particulier des signes contraires).

Instructions à adressage direct

Ce sont des instructions qui contiennent une adresse opérande complète de 16 bits (deux octets) et ont par conséquent une longueur totale de trois octets. Attention ! Comme déjà maintes fois signalé, *le troisième octet est la moitié la plus significative de l'adresse.*

Le format général du premier octet est le suivant :
001XX010, la correspondance de XX étant la suivante :

00 pour SHLD
01 pour LHLD
10 pour STA
11 pour LDA

Quelques mots de commentaire sur les modes d'adressage paraissent indiqués ici. Pendant de nombreuses années, l'adressage direct — avec par conséquent une adresse, de longueur suffisante pour adresser toute la mémoire, incluse dans l'instruction — a été le mode d'adressage privilégié : il présente l'avantage évident de la clarté et de la simplicité. Cependant, les tailles de mémoire augmentant régulièrement, l'encombrement d'adresses longues a commencé à se faire sentir, tout particulièrement lorsque les concepteurs de la série IBM 360 décidèrent qu'un jour les mémoires centrales pourraient atteindre 16 millions d'octets. Des adresses de 24 bits, une par instruction en moyenne, c'était beaucoup, d'où l'idée des registres de base et des adresses courtes dans les instructions ne permettant qu'un *déplacement* modéré de l'ordre de 1/4 de K à 4 K selon les machines. Nous vous recommandons à ce sujet de voir les articles **adressage** dans l'Encyclopédie, (*L'Informatique nouvelle* n° 45 à 48, novembre 73 à février 1974), et surtout de lire le remarquable article de John Michels entièrement consacré aux choix de modes d'adressage lors de la conception d'une machine, sous le

Les instructions du 8080 rangées par groupes, avec leur code machine en hexa. A partir de « pseudo instructions », commandes de l'assembleur dont nous ne parlons pas en détail dans le texte

JUMP

C3 JMP
C2 JNZ
CA JZ
D2 JNC
DA JC
E2 JPO
EA JPE
F2 JP
FA JM
} Adr
E9 PCHL

CALL

CD CALL
C4 CNZ
CC CZ
D4 CNC
DC CC
E4 CPO
EC CPE
F4 CP
FC CM
} Adr

RETURN

C9 RET
C0 RNZ
C8 RZ
D0 RNC
D8 RC
E0 RPO
E8 RPE
F0 RP
F8 RM

RESTART

C7 RST 0
CF RST 1
D7 RST 2
DF RST 3
E7 RST 4
EF RST 5
F7 RST 6
FF RST 7

ROTATE

07 RLC
0F RRC
17 RAL
1F RAR

CONTROL

00 NOP
76 HLT
F3 DI
FB EI

MOVE (cont)

58 MOV E,B
59 MOV E,C
5A MOV E,D
5B MOV E,E
5C MOV E,H
5D MOV E,L
5E MOV E,M
5F MOV E,A

60 MOV H,B
61 MOV H,C
62 MOV H,D
63 MOV H,E
64 MOV H,H
65 MOV H,L
66 MOV H,M
67 MOV H,A

68 MOV L,B
69 MOV L,C
6A MOV L,D
6B MOV L,E
6C MOV L,H
6D MOV L,L
6E MOV L,M
6F MOV L,A

70 MOV M,B
71 MOV M,C
72 MOV M,D
73 MOV M,E
74 MOV M,H
75 MOV M,L
---
77 MOV M,A

78 MOV A,B
79 MOV A,C
7A MOV A,D
7B MOV A,E
7C MOV A,H
7D MOV A,L
7E MOV A,M
7F MOV A,A

ACCUMULATOR*

80 ADD B
81 ADD C
82 ADD D
83 ADD E
84 ADD H
85 ADD L
86 ADD M
87 ADD A

88 ADC B
89 ADC C
8A ADC D
8B ADC E
8C ADC H
8D ADC L
8E ADC M
8F ADC A

90 SUB B
91 SUB C
92 SUB D
93 SUB E
94 SUB H
95 SUB L
96 SUB M
97 SUB A

98 SBB B
99 SBB C
9A SBB D
9B SBB E
9C SBB H
9D SBB L
9E SBB M
9F SBB A

A0 ANA B
A1 ANA C
A2 ANA D
A3 ANA E
A4 ANA H
A5 ANA L
A6 ANA M
A7 ANA A

A8 XRA B
A9 XRA C
AA XRA D
AB XRA E
AC XRA H
AD XRA L
AE XRA M
AF XRA A

B0 ORA B
B1 ORA C
B2 ORA D
B3 ORA E
B4 ORA H
B5 ORA L
B6 ORA M
B7 ORA A

B8 CMP B
B9 CMP C
BA CMP D
BB CMP E
BC CMP H
BD CMP L
BE CMP M
BF CMP A

MOVE IMMEDIATE

06 MVI B,
0E MVI C,
16 MVI D,
1E MVI E,
26 MVI H,
2E MVI L,
36 MVI M,
3E MVI A,
} D8

Acc IMMEDIATE*

C6 ADI
CE ACI
D6 SUI
DE SBI
E6 ANI
EE XRI
F6 ORI
FE CPI
} D8

LOAD IMMEDIATE

01 LXI B,
11 LXI D,
21 LXI H,
31 LXI SP,
} D16

STACK OPS

C5 PUSH B
D5 PUSH D
E5 PUSH H
F5 PUSH PSW

C1 POP B
D1 POP D
E1 POP H
F1 POP PSW*

E3 XTHL
F9 SPHL

DOUBLE ADD†

09 DAD B
19 DAD D
29 DAD H
39 DAD SP

SPECIALS

EB XCHG
27 DAA*
2F CMA
37 STC†
3F CMC†

LOAD/STORE

0A LDAX B
1A LDAX D
2A LHLD Adr
3A LDA Adr

02 STAX B
12 STAX D
22 SHLD Adr
32 STA Adr

INPUT/OUTPUT

D3 OUT
DB IN
} D8

MOVE

40 MOV B,B
41 MOV B,C
42 MOV B,D
43 MOV B,E
44 MOV B,H
45 MOV B,L
46 MOV B,M
47 MOV B,A

48 MOV C,B
49 MOV C,C
4A MOV C,D
4B MOV C,E
4C MOV C,H
4D MOV C,L
4E MOV C,M
4F MOV C,A

50 MOV D,B
51 MOV D,C
52 MOV D,D
53 MOV D,E
54 MOV D,H
55 MOV D,L
56 MOV D,M
57 MOV D,A

INCREMENT**

04 INR B
0C INR C
14 INR D
1C INR E
24 INR H
2C INR L
34 INR M
3C INR A

03 INX B
13 INX D
23 INX H
33 INX SP

DECREMENT**

05 DCR B
0D DCR C
15 DCR D
1D DCR E
25 DCR H
2D DCR L
35 DCR M
3D DCR A

0B DCX B
1B DCX D
2B DCX H
3B DCX SP

PSEUDO INSTRUCTION

ORG Adr
END
EQU D16
SET D16
DS D16
DB D8 [ ]
DW D16 [ ]

IF D16
ENDIF

MACRO [ ]
ENDM

CONSTANT DEFINITION

0BDH, 1AH — Hex
105D, 105 — Decimal
72O, 72Q — Octal
11011B, 00110B — Binary
'TEST', 'A', 'B' — ASCII

OPERATORS

(,)
*, /, MOD, SHL, SHR
+, –
NOT
AND
OR, XOR

STANDARD SETS

A SET 7
B SET 0
C SET 1
D SET 2
E SET 3
H SET 4
L SET 5
M SET 6
SP SET 6
PSW SET 6

FLAG BYTE STACK FORMAT

7	6	5	4	3	2	1	Ø
S	Z	Ø	A	C	Ø	P	1	C

D8 = constante, ou expression arithmétique ou logique, ayant un longueur d'un octet (8 bits)

D16 = constante, ou expression arithmétique ou logique, ayant une longueur de deux octets (16 bits)

Adr = adresse de 16 bits

* = tous les drapeaux (CY, Z, S, P, AC) sont affectés

† = seul le drapeau C est affecté

** = tous les drapeaux, sauf C, sont affectés (exception : INX et DCX n'affectent aucun drapeau)

titre «le méga-mini succède au modèle T» (*L'Informatique nouvelle* n° 59 février 1975). Malheureusement, ce «perfectionnement», le système de base et déplacement, nous ravale à peu près dans la pratique au niveau de la pagination des minis les plus affreux.

C'est le revers de la médaille, celui que connaissent les utilisateurs des petits modèles de 360/370 (eux, ou leurs compilateurs), sacrifiés aux possibilités des gros modèles du haut de gamme : le malheur des uns fait le bonheur des autres...

C'est donc une satisfaction de voir la taille de mémoire centrale de 64K octets prévue pour le 8080 (ce qui est large, rappelons-nous que le 360/30 allait aussi à 64K dans les débuts, et que plus près de nous les études de marché menées par Signetics pour le PIP 2650 ont montré que plus de 90 % des systèmes auraient moins de 32K). L'adresse s'inscrit alors joliment dans deux octets, ce qui a permis de doter les machines d'une quantité appréciable d'instructions à adressage direct, et en particulier *toutes les instructions fondamentales :*

— chargement et déchargement de l'accu et de HL,

— chargement de B, D, H, SP,

— branchements conditionnels et inconditionnels,

— appels de sous-programmes.

On le voit — et c'est bien encore là l'informatique à l'envers — on bénéficie avec Alcyane d'un retour à une netteté de programmation un peu oubliée.

Voyons maintenant une à une les instructions à adressage direct de chargement ou de déchargement (les autres sont traitées dans leurs sections respectives).

STA *Store Accumulator Direct* (stocker l'accumulateur en adressage direct).

00110010 soit **32** suivi des deux octets de l'adresse.

Le contenu de l'accu remplace celui de l'octet situé à l'adresse mémoire indiquée dans l'instruction. Aucun drapeau n'est affecté, et l'accumulateur reste inchangé bien entendu.

Liste des instructions dans l'ordre numérique des codes opérations, suivie d'une table de correspondance hexa-ASCII. (Voir l'Encyclopédie dans L'Informatique nouvelle, *n° de février 76).*

00	NOP		2B	DCX	H	56	MOV	D,M	81	ADD	C	AC	XRA	H	D7	RST	2						
01	LXI	B,D16	2C	INR	L	57	MOV	D,A	82	ADD	D	AD	XRA	L	D8	RC							
02	STAX	B	2D	DCR	L	58	MOV	E,B	83	ADD	E	AE	XRA	M	D9	- - -							
03	INX	B	2E	MVI	L,D8	59	MOV	E,C	84	ADD	H	AF	XRA	A	DA	JC	Adr						
04	INR	B	2F	CMA		5A	MOV	E,D	85	ADD	L	B0	ORA	B	DB	IN	D8						
05	DCR	B	30	- - -		5B	MOV	E,E	86	ADD	M	B1	ORA	C	DC	CC	Adr						
06	MVI	B,D8	31	LXI	SP,D16	5C	MOV	E,H	87	ADD	A	B2	ORA	D	DD	- - -							
07	RLC		32	STA	Adr	5D	MOV	E,L	88	ADC	B	B3	ORA	E	DE	SBI	D8						
08	- - -		33	INX	SP	5E	MOV	E,M	89	ADC	C	B4	ORA	H	DF	RST	3						
09	DAD	B	34	INR	M	5F	MOV	E,A	8A	ADC	D	B5	ORA	L	E0	RPO							
0A	LDAX	B	35	DCR	M	60	MOV	H,B	8B	ADC	E	B6	ORA	M	E1	POP	H						
0B	DCX	B	36	MVI	M,D8	61	MOV	H,C	8C	ADC	H	B7	ORA	A	E2	JPO	Adr						
0C	INR	C	37	STC		62	MOV	H,D	8D	ADC	L	B8	CMP	B	E3	XTHL							
0D	DCR	C	38	- - -		63	MOV	H,E	8E	ADC	M	B9	CMP	C	E4	CPO	Adr						
0E	MVI	C,D8	39	DAD	SP	64	MOV	H,H	8F	ADC	A	BA	CMP	D	E5	PUSH	H						
0F	RRC		3A	LDA	Adr	65	MOV	H,L	90	SUB	B	BB	CMP	E	E6	ANI	D8						
10	- - -		3B	DCX	SP	66	MOV	H,M	91	SUB	C	BC	CMP	H	E7	RST	4						
11	LXI	D,D16	3C	INR	A	67	MOV	H,A	92	SUB	D	BD	CMP	L	E8	RPE							
12	STAX	D	3D	DCR	A	68	MOV	L,B	93	SUB	E	BE	CMP	M	E9	PCHL							
13	INX	D	3E	MVI	A,D8	69	MOV	L,C	94	SUB	H	BF	CMP	A	EA	JPE	Adr						
14	INR	D	3F	CMC		6A	MOV	L,D	95	SUB	L	C0	RNZ		EB	XCHG							
15	DCR	D	40	MOV	B,B	6B	MOV	L,E	96	SUB	M	C1	POP	B	EC	CPE	Adr						
16	MVI	D,D8	41	MOV	B,C	6C	MOV	L,H	97	SUB	A	C2	JNZ	Adr	ED	- - -							
17	RAL		42	MOV	B,D	6D	MOV	L,L	98	SBB	B	C3	JMP	Adr	EE	XRI	D8						
18	- - -		43	MOV	B,E	6E	MOV	L,M	99	SBB	C	C4	CNZ	Adr	EF	RST	5						
19	DAD	D	44	MOV	B,H	6F	MOV	L,A	9A	SBB	D	C5	PUSH	B	F0	RP							
1A	LDAX	D	45	MOV	B,L	70	MOV	M,B	9B	SBB	E	C6	ADI	D8	F1	POP	PSW						
1B	DCX	D	46	MOV	B,M	71	MOV	M,C	9C	SBB	H	C7	RST	0	F2	JP	Adr						
1C	INR	E	47	MOV	B,A	72	MOV	M,D	9D	SBB	L	C8	RZ		F3	DI							
1D	DCR	E	48	MOV	C,B	73	MOV	M,E	9E	SBB	M	C9	RET	Adr	F4	CP	Adr						
1E	MVI	E,D8	49	MOV	C,C	74	MOV	M,H	9F	SBB	A	CA	JZ		F5	PUSH	PSW						
1F	RAR		4A	MOV	C,D	75	MOV	M,L	A0	ANA	B	CB	- - -		F6	ORI	D8						
20	- - -		4B	MOV	C,E	76	HLT		A1	ANA	C	CC	CZ	Adr	F7	RST	6						
21	LXI	H,D16	4C	MOV	C,H	77	MOV	M,A	A2	ANA	D	CD	CALL	Adr	F8	RM							
22	SHLD	Adr	4D	MOV	C,L	78	MOV	A,B	A3	ANA	E	CE	ACI	D8	F9	SPHL							
23	INX	H	4E	MOV	C,M	79	MOV	A,C	A4	ANA	H	CF	RST	1	FA	JM	Adr						
24	INR	H	4F	MOV	C,A	7A	MOV	A,D	A5	ANA	L	D0	RNC		FB	EI							
25	DCR	H	50	MOV	D,B	7B	MOV	A,E	A6	ANA	M	D1	POP	D	FC	CM	Adr						
26	MVI	H,D8	51	MOV	D,C	7C	MOV	A,H	A7	ANA	A	D2	JNC	Adr	FD	- - -							
27	DAA		52	MOV	D,D	7D	MOV	A,L	A8	XRA	B	D3	OUT	D8	FE	CPI	D8						
28	- - -		53	MOV	D,E	7E	MOV	A,M	A9	XRA	C	D4	CNC	Adr	FF	RST	7						
29	DAD	H	54	MOV	D,H	7F	MOV	A,A	AA	XRA	D	D5	PUSH	D									
2A	LHLD	Adr	55	MOV	D,L	80	ADD	B	AB	XRA	E	D6	SUI	D8									

HEX-ASCII TABLE

00	NUL		2B	+		56	V		
01	SOH		2C	,		57	W		
02	STX		2D	-		58	X		
03	ETX		2E	.		59	Y		
04	EOT		2F	/		5A	Z		
05	ENQ		30	0		5B	[		
06	ACK		31	1		5C	\		
07	BEL		32	2		5D	]		
08	BS		33	3		5E	^	(↑)	
09	HT		34	4		5F	—	(←)	
0A	LF		35	5		60	`		
0B	VT		36	6		61	a		
0C	FF		37	7		62	b		
0D	CR		38	8		63	c		
0E	SO		39	9		64	d		
0F	SI		3A	:		65	e		
10	DLE		3B	;		66	f		
11	DC1	(X-ON)	3C	<		67	g		
12	DC2	(TAPE)	3D	=		68	h		
13	DC3	(X-OFF)	3E	>		69	i		
14	DC4	~~(TAPE)~~	3F	?		6A	j		
15	NAK		40	@		6B	k		
16	SYN		41	A		6C	l		
17	ETB		42	B		6D	m		
18	CAN		43	C		6E	n		
19	EM		44	D		6F	o		
1A	SUB		45	E		70	p		
1B	ESC		46	F		71	q		
1C	FS		47	G		72	r		
1D	GS		48	H		73	s		
1E	RS		49	I		74	t		
1F	US		4A	J		75	u		
20	SP		4B	K		76	v		
21	!		4C	L		77	w		
22	"		4D	M		78	x		
23	#		4E	N		79	y		
24	$		4F	O		7A	z		
25	%		50	P		7B	{		
26	&		51	Q		7C	|		
27	'		52	R		7D	}	(ALT MODE)	
28	(		53	S		7E	~		
29	)		54	T		7F	DEL	(RUB OUT)	
2A	*		55	U					

D 8 = constante, ou expression arithémitique ou logique, ayant une longueur d'un octet (8 bits)

D 16 = constante, ou expression arithémitique ou logique, ayant une longueur de deux octets (16 bits)

Adr = adresse de 16 bits

Exemple : chacune des instructions suivantes stocke le contenu de l'accu en 05B3H :

assembleur		absolu
code	adresse opérande	une seule traduction
STA	5B3H	en langage machine :
STA	1459	32B305
STA	0101101100011B	

On voit qu'en assembleur on a le choix de trois modes d'écriture pour un nombre : en hexa, signalé par H, en décimal sans rien ajouter, et en binaire signalé par B. Après assemblage, chacune de ces instructions donnera la même instruction en langage machine, bien entendu : 32B305 en hexa. Si l'on préfère le binaire, on peut l'écrire sur la feuille

00110010
10110011
00000101

ce qui est bigrement moins compact que l'hexa. Cependant, nous avons remarqué que ce dernier est craint par beaucoup, qui se privent ainsi d'une représentation quand même très pratique pour les machines à octets. Là où on est au pied du mur c'est quand il faut, à partir du programme écrit en hexa, entrer les octets un à un au pupitre d'Alcyane : il faut acquérir le réflexe établissant la correspondance entre le chiffre hexa et la configuration de quatre bits équivalente, sans avoir à recalculer mentalement.

Au risque de faire sourire les chevronnés, nous allons une fois de plus donner quelques «trucs» mnémoniques qui pourront faciliter l'opération au début, puis très vite vous vous apercevrez que vous n'avez même plus besoin de vous en servir.

Pour les chiffres de zéro à sept, c'est la même chose qu'en octal, et donc très facile (vous voyez ! s'exclament les partisans de l'octal...). Nous voici donc en face de nos quatre voyants. Zéro, tout est éteint. Un, la diode de droite est allumée avec son petit œil rouge. Deux, c'est la deuxième. Trois, les deux de droite. Quatre ça fait cent (horreur mathématique, on le dit quand même), cinq 101, six 110 et sept, bien caractéristique, les trois lampes de droite sont rouges, celle de gauche éteinte. Fin de l'octal.

Entrons dans l'hexa maintenant : il faut mémoriser huit autres combinaisons. Huit : la diode de gauche est allumée toute seule, comme dans tout chiffre rond binaire (1, 2, 4, 8, 16, etc.). Neuf : un

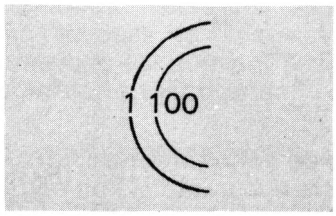
œil rouge à chaque bout pour le huit plus un. A partir de là, c'est le grand jeu, les six chiffres-lettres de A à F. Rappelons d'abord leur valeur :

décimal	chiffre hexa	binaire
dix	A	1010
onze	B	1011
douze	C	1100
treize	D	1101
quatorze	E	1110
quinze	F	1111

Par pure coïncidence, on voit apparaître deux fois «dix» dans les structures de A : ce «dix-dix» est bien caractéristique pour idendifier le dix hexa A. A l'autre bout, le dernier chiffre F allume toutes les diodes électro-luminescentes, et se reconnaît d'emblée quand on explore des octets mémoire entièrement remplis de bits un.

Entre les deux, restent quatre chiffres intermédiaires. Nous allons vous proposer quelques mnémoniques, pas bien fameux... surtout pour susciter les vôtres. B, D et E comportent chacun un seul zéro : celui de B est en deuxième position (c'est la deuxième lettre de l'alphabet), pour D il est un cran plus loin et pour E il pend comme un œuf (euh, euh !) tout au bout...

Quant à C, il peut évoquer le dessin de la lettre :



Tout ceci peut être considéré comme débile ! Alors, fermez les yeux sur ce passage en ayant la charité de l'oublier... Mais peut-être l'œuf ou le C vous serviront-ils quand même par moment ? Il y a toujours un de ces caractères un peu rétif...

LDA *Load Accumulator Direct* (charger l'accumulateur en adressage direct).

00111010 soit **3A** suivi des deux octets de l'adresse (toujours inversés).

Le contenu de l'octet mémoire désigné par l'adresse vient remplacer le contenu de l'accu. Aucun flag n'est affecté, et l'octet en mémoire reste inchangé.

Exemple : les trois instructions suivantes jouent le même rôle, c'est-à-dire charger dans l'accu le contenu de l'octet situé en 300H :

assembleur	absolu
LDA 300H	une seule traduction
LDA 3x(16x16)	en langage machine :
LDA 200H + 256	3A0003

On voit d'autres fantaisies permises par l'assembleur : opérations arithmétiques pour calculer l'adresse opérande, mixage des systèmes de numération... Ne nous étonnons plus qu'il ait un encombrement mémoire de 8K !

SHLD *Store H and L Direct* (stocker H et L en adressage direct)
00100010 soit **22** et les deux octets de l'adresse A.

Le contenu de L est stocké en mémoire à l'adresse A indiquée, et celui de H à l'adresse immédiatement supérieure, A + 1. On notera l'inversion classique des deux moitiés de l'adresse. Aucun drapeau n'est affecté, H et L restent inchangés. *Exemple* : il y a AEH dans H et 29H dans L. L'instruction

SHLD 10AH

va provoquer en mémoire les changements suivants (tout est en hexa) :

adresse mémoire	contenu mémoire avant SHLD	contenu mémoire après SHLD
109	00	00
10A	00	29
10B	00	AE
10C	00	00

LHLD *Load H and L Direct* (charger H et L en adressage direct)
00101010 soit **2A** et les deux octets de l'adresse A

Le contenu de l'octet mémoire situé à l'adresse A est transféré dans L, et celui de l'octet en A + 1 va dans H. Aucun flag affecté, contenu mémoire inchangé.

Le fonctionnement de cette instruction est exactement symétrique de SHLD, en particulier en ce qui concerne l'inversion. Si donc on a stocké temporairement le contenu de HL en mémoire par un SHLD ADSTK, un LHLD ADSTK le récupérera en bon ordre dans H et L, sans surprise.

Instructions de branchement

... ou instructions de saut, ou instructions de rupture de séquence, selon le vocabulaire que vous affectionnez. Il en existe une bonne variété, conditionnelles ou inconditionnelles, toutes en adressage direct (sauf une). Nous verrons plus

loin les instructions d'appel de sous-programmes et de retour, ainsi que l'instruction Restart pour les interruptions : ces instructions sont apparentées aux instructions de branchement, puisqu'elles aussi font quitter le mode normal de déroulement séquentiel du programme.

Commençons par la seule instruction de branchement longue d'un octet (toutes les autres, en adressage direct, sont naturellement longues de trois octets) :

PCHL *Load Program Counter* (charger le compteur de programme)
11101001 soit **E9**

C'est une instruction de branchement inconditionnel indirect. Le contenu de HL est chargé dans PC (le contenu de H, high, remplace les huit bits de gauche du compteur ordinal et celui de L, low, les huit bits de droite). L'instruction suivante exécutée par la machine sera donc celle située à l'adresse contenue dans HL. Aucun drapeau n'est affecté, et le contenu de HL reste inchangé.

Exemple 1 : si H contient 71H et L contient A3H, l'instruction PCHL effectuera un branchement à l'adresse 71A3H.

Exemple 2 :

adresse mémoire	label	code	opérande	langage machine
:				
43C5	ADR :	DW	LIEU	B3
				44
:				:
44AA	DEPA :	LHLD	ADR	2A
44AD		PCHL		C5
:				43
				E9
				:
44B3	LIEU :	NOP		00
:				:

Dans cette séquence de programme on voit une des *pseudo-instructions* de l'assembleur (qui n'ont pas d'équivalent en langage machine, où elles sont inutiles car dispose soi-même ce que l'on veut, où l'on veut, dans la mémoire). C'est une *définition de constante* DW (Define Word, définir un mot) qui permet de placer dans deux octets consécutifs une information spécifiée dans la zone opérande du DW. Si l'on y met C437H, l'assembleur mettra C4 dans le premier octet et 37 dans le deuxième. Ici nous avons écrit LIEU : l'assembleur cherche dans la table des labels (étiquettes, si vous préférez) la valeur numérique de LIEU et la met avec inversion dans les deux octets du DW. Si l'on travaille en langage machine — avec écriture préalable en assembleur comme nous l'avons conseillé — on suit la règle :

— une adresse logée en mémoire doit être inversée, qu'elle soit dans une constante (style DW) ou dans la zone adresse opérande d'une instruction à adressage direct ou à opérande immédiat,
— une adresse logée dans une paire de registres (HL, PC, etc.) doit être dans le sens normal : l'inversion est réalisée automatiquement au chargement et déchargement.

Mais revenons-en à notre séquence de programme. Le LHLD charge le contenu de 43C5 et 43C6 dans HL avec inversion, 44 dans H et B3 dans L. Alors PCHL agit, et met 44B3 dans PC, ce qui effectue le branchement à cette adresse. L'instruction suivante exécutée sera donc le bouche-trou NOP, le code opération 00, sans doute début d'une séquence plus intéressante !

JMP *Jump* (saut, sauter)

C'est la première du groupe des instructions de branchement à adressage direct, donc longues de trois octets. Avant de la détailler, quelques mots sur la structure des codes opération des instructions de ce groupe. Ils se présentent comme suit : **11XXX01Y** suivi des deux octets de l'adresse, et :

— Y vaut 1 pour JMP, et zéro pour les huit instructions *conditionnelles* de branchement,
— XXX vaut

000 pour JMP et JNZ	100 pour JPO
001 pour JZ	101 pour JPE
010 pour JNC	110 pour JP
011 pour JC	111 pour JM

A l'examen de cette structure, on constate que sa logique reste apparente si l'on écrit en octal, et qu'elle est cachée en hexa. En effet, les codes en octal seront 303, 302, 312, 332, 342, 352, 362 et 372. En hexa nous aurons C3, C2, CA, D2, DA, E2, EA, F2 et FA. Nous avons mis là le doigt sur ce qui, en pratique, est le seul avantage de l'écriture en octal pour le langage machine du 8080 ; cet avantage nous semble cependant très marginal. Et par contre une zone où il n'y a que des octets remplis de bits un donne en octal

377
377
377

alors qu'en hexa nous avons :

FF
FF
FF

ce qui est incontestablement plus clair et homogène.

Mais revenons à la description de JMP : l'instruction provoque un branchement inconditionnel, c'est-à-dire que l'instruction suivante exécutée sera celle placée à l'adresse mémoire figurant dans le JMP.

Exemple :

	assembleur			absolu
adresse mémoire	label	code	opérande	
:				
3C00		JMP	RAZ	C3003E
3C03	AD :	ADI	2	C602
:				:
3D00	CHAR :	MVI	A,3	3E03
3D02		JMP	3C03H	C3033C
:				:
3E00	RAZ :	XRA	A	AF
3E01		JMP	$—101H	C3003D
:				:

Décortiquons ce bout de programme. L'instruction JMP RAZ charge 3E00 dans PC, ce qui sera donc l'adresse de la prochaine instruction : on exécute donc ensuite XRA A qui remet l'accu à zéro. Le JMP qui suit a une zone opérande plutôt bizarre. Qu'est-ce que c'est que ce signe dollar dont on soustrait 101H ? Un coup d'œil sur le listing machine nous renseigne : l'adresse dans le JMP assemblé vaut 3D00, et 3D00 + 101 = 3E01, c'est-à-dire l'adresse de l'instruction elle-même. Le signe $ est donc en assembleur 8080 le symbole indiquant l'adresse de l'instruction dans laquelle on le trouve (c'est souvent l'astérisque qui est utilisée pour cette fonction dans d'autres assembleurs).

Notre JMP nous ramène donc sur le label CHARgement qui repère un MVI, lequel charge effectivement 3 dans l'accu, après quoi le JMP nous branche en 3C03 pour effectuer l'addition immédiate de 2 dans A, et les instructions à partir de 3C05 se déroulent ensuite...

JC *Jump if carry* (sauter si report)
11011010 soit **DA** suivi des deux octets de l'adresse.

Si le drapeau de report CY est égal à un, le branchement s'effectue à l'adresse contenue dans l'instruction. Si CY = 0, le programme continue en séquence normale : (PC) + 1 → PC. Aucun drapeau de condition n'est affecté.

On notera que toutes les instructions de branchement conditionnel fonctionnent sur le même principe : si la condition est satisfaite, le branchement s'effectue. Les exemples viendront plus loin.

JNC *Jump if no carry* (sauter si pas de report)
11010010 soit **D2** suivi des deux octets de l'adresse.

Si le bit de report CY est égal à zéro, le branchement s'effectue. Sinon, séquence. Aucun flag affecté.

JZ *Jump if zero* (sauter si zéro)
11001010 soit **CA** suivi des deux octets de l'adresse.

Si le drapeau Z est égal à un (c'est-à-dire si le contenu de l'accu est nul), le saut s'effectue. Sinon, séquence. Aucun flag n'est affecté.

JNZ *Jump if not zero* (sauter si non zéro)
11000010 soit **C2** suivi des deux octets de l'adresse.

Si le bit 2 est à zéro (contenu de l'accu non nul), le branchement s'effectue. Sinon, on continue en séquence. Aucun flag n'est affecté.

JM *Jump if minus* (sauter si négatif)
11111010 soit **FA** suivi des deux octets de l'adresse.

Si le flag de signe S est à un (indiquant un résultat négatif), le saut s'effectue. Sinon, on continue en séquence. Aucun flag affecté.

JPO *Jump if parity odd* (sauter si parité impaire)
11100010 soit **E2** suivi des deux octets de l'adresse.

Si le bit de parité P est à zéro (indiquant un nombre de bits 1 impair dans A), le saut s'effectue. Sinon, séquence. Aucun drapeau affecté.

Exemple : nous allons voir trois méthodes de branchement différentes, mais fonctionnellement équivalentes, pour choisir de sauter vers un emplacement parmi deux en fonction du signe d'un nombre. Le nombre (un octet) a été placé dans le registre C.

	assembleur		absolu
label	code	opérande	
UN :	MOV	A, C	79
	ANI	80H	E680
	JZ	PLUS	CAXXXX
	JNZ	MOINS	C2XXXX
DEUX :	MOV	A, C	79
	RLC		07
	JNC	PLUS	D2XXXX
	JMP	MOINS	C3XXXX
TROIS :	MOV	A, C	79
	ADI	0	C600
	JM	MOINS	FAXXXX
PLUS :	.....		

L'instruction de ET immédiat dans le bloc UN met à zéro tous les bits de l'octet donné, à l'exception du bit de signe qui n'est pas modifié. Si ce bit de signe était zéro, le drapeau de condition Z se trouvera alors mis à un, et l'instruction JZ provoquera un branchement vers le bloc de traitement PLUS.

Dans le cas contraire, l'instruction JZ ne fera qu'incrémenter le PC de trois, et on exécutera l'instruction JNZ qui provoquera un branchement

vers le bloc de traitement PLUS (rappelons en effet que le flag Z n'est pas affecté par les instructions de branchement : JZ n'a donc pas modifié son état, et JNZ peut valablement l'utiliser).

L'instruction RLC dans le bloc DEUX fait prendre au flag CY la valeur du bit de signe de l'octet qui vient d'être placé dans A (en effet le MOV, rappelons-le, n'a aucune action sur les flags qui avaient donc toujours un état déterminé par quelque opération antérieure, et on ne peut donc tester directement le bit de signe). Si ce bit de signe est à zéro, l'instruction JNC lors de son exécution provoquera un branchement vers PLUS. Sinon, on passera à l'instruction JMP qui, elle, provoquera un branchement vers MOINS (remarquons que nous aurions pu mettre JC à la place de JMP, le résultat atteint étant exactement le même dans ce cas).

L'instruction d'addition immédiate de zéro dans le bloc TROIS ne provoque évidemment aucun changement dans le contenu de 'accu. Par contre, étant donné que ADI affecte les cinq flags, ils prendront tous individuellement à ce moment là la valeur correspondant au dit contenu. Si le bit de signe est à un, S y sera mis également et JM provoquera un branchement vers MOINS. Sinon, le programme continuera en séquence et on commencera donc l'exécution du bloc PLUS.

La leçon à tirer de ces trois séquences est que, une instruction de saut conditionnelle exploitant le positionnement des flags, il faut le provoquer effectivement avant qu'elle ne s'exécute. Ce positionnement est usuellement à réaliser juste en amont, mais il peut également l'être bien plus tôt, à condition qu'aucune instruction affectant les flags ne soit exécutée ensuite : on peut donc exploiter «à distance» le résultat d'une opération, et ceci à plusieurs reprises, les instructions de branchement n'affectant pas les flags qui conservent ainsi leur valeur initiale.

3 - 8 - 5. Instructions pour les sous-programmes

Ce sont des instructions particulièrement pratiques, qui combinent le branchement (conditionnel ou non) vers un sous-programme et le stockage du PC sur pile, pour un retour automatique vers le programme appelant aux moyens des instructions de retour qu'on verra plus loin. Le fonctionnement de ces instructions en trois octets compte parmi les plus complexes du 8080, c'est la raison — on s'en rappelle — pour laquelle nous avons choisi l'instruction CALL pour une description très détaillée de l'enchaînement cycles machine/ états, accompagnée des chronogrammes des principaux signaux du bus d'Alcyane.

Les instructions de cette catégorie ont un code opération présentant la structure suivante :

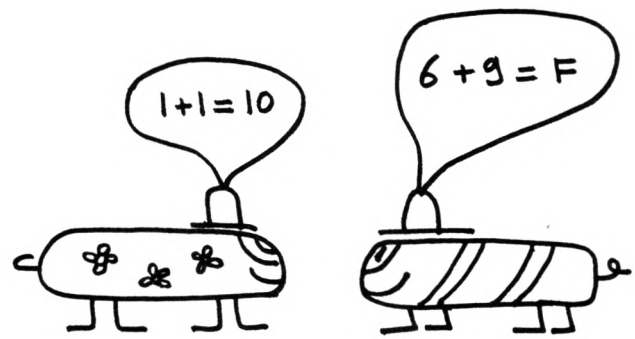
11XXX10Y suivi des deux octets de l'adresse.

La ressemblance avec les JMP est nette (10 au lieu de 01), et de même :

— Y vaut 1 pour CALL, et zéro pour les huit instructions d'appel *conditionnel* de sous-programmes
— XXX vaut

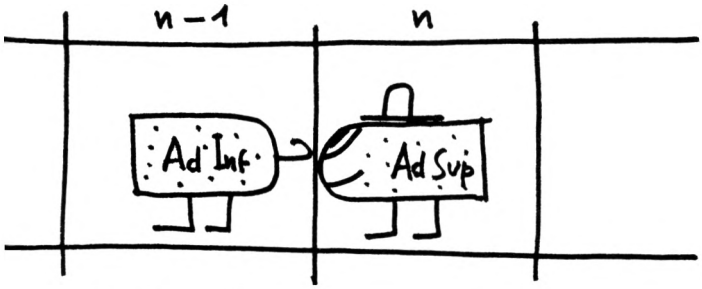
000 pour CNZ	100 pour CPO
001 pour CZ ou CALL	101 pour CPE
010 pour CNC	110 pour CP
011 pour CC	111 pour CM

On remarquera une curieuse dissonance en se reportant à la liste des codes JMP, et du reste nous avons constaté que certaines brochures du μP sont fausses pour la liste des codes des CALL, car elles respectent le parallélisme strict (un décalage arithmétique de 4 — 2 = 2 entre les codes opération comme on le voit facilement : 100 — 10 = 10 en langage Gibi).

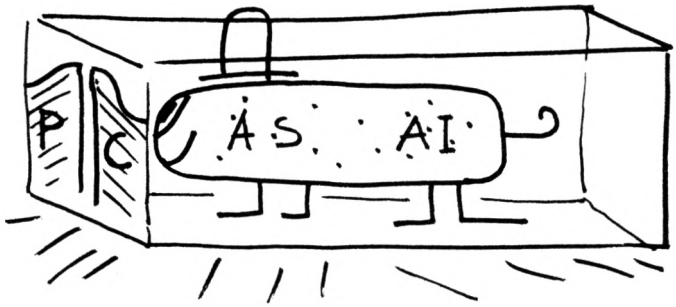


Naturellement, comme dans toutes les instructions à adressage direct, les deux moitiés de l'adresse sont interverties dans les instructions CALL, puisque c'est de cette façon que les adresses doivent toujours être stockées en mémoire.

Rappelons au passage qu'en anglais CALL signifie *appeler, appel*. Ces instructions «appellent» donc des sous-programmes (appellent = provoquent un branchement vers) en fonction de conditions spécifiées. Si la condition est vraie, l'adresse pour le retour est stockée sur le sommet de la pile, le contenu de SP au cours de cette opération étant décrémenté de deux unités (la pile se développe en remontant vers les adresses faibles), puis le deuxième octet de l'instruction est chargé dans la moitié droite de PC, et le troisième dans sa moitié gauche : le branchement s'effectue alors. On peut visualiser cette opération par le Poste de Commandement du μP, n'ayant qu'une entrée par laquelle les deux moitiés supérieure et inférieure du Gibi-adresse passent l'une après l'autre : cette disposition particulière d'accès est cause de l'inversion en mémoire, et pour cela même rétablit l'ordre normal au chargement des registres.



Gibi adresse en mémoire



Gibi adresse dans un registre

Si la condition spécifiée est fausse, le programme continue de se dérouler en séquence, et on exécute l'instruction qui suit immédiatement le CALL. Voici la description des instructions du groupe (on trouvera des exemples de programmation plus loin) : aucune d'entre elles n'affecte les drapeaux.

CALL *Call* (appel)
11001101 soit **CD** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue toujours (appel inconditionnel).

CC *Call if Carry* (appel si report)
11011100 soit **DC** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue si le flag CY est à un.

CNC *Call if no Carry* (appel si pas de report)
11010100 soit **D4** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue si le drapeau CY est à zéro.

CZ *Call if Zero* (appel si zéro)
11001100 soit **CC** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue si le flag Z est à un (c'est-à-dire si une opération a préalablement donné un résultat zéro dans l'accumulateur). Attention à ce comportement bizarre du Z dû à une coïncidence pour lui entre la nature de la condition testée (zéro ou non) et la nature même d'un drapeau (un ou zéro). En programmation tout ceci est du reste transparent : on écrit l'opération, et on met CZ, CNZ ou autre à la suite sans avoir à se poser de questions en général.

CNZ *Call if Not Zero* (appel si non zéro)
11000100 soit **C4** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue si le flag Z est à zéro, c'est-à-dire si l'accu est non nul (voir remarques instruction précédente).

CM *Call if Minus* (appel si moins)
11111100 soit **FC** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue si le drapeau S est à un, indiquant un résultat négatif dans l'accu.

CP *Call if Plus* (appel si plus)
11110100 soit **F4** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue si le bit S est à zéro, indiquant un résultat positif dans l'accu.

CPE *Call if Parity Even* (appel si parité paire)
11101100 soit **EC** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue si le flag P est à un, indiquant une parité paire.
Rappelons que la parité en question est celle du *nombre de bits un* d'un résultat d'opération obtenu dans l'accu.

CPO *Call if Parity Odd* (appel si parité impaire)
11100100 soit **E4** suivi des deux octets de l'adresse du sous-programme.
L'appel du sous-programme s'effectue si le bit P est à zéro, indiquant une parité impaire (voir rappel instruction précédente).

Des instructions symétriques de celles que nous venons de voir permettent le retour automatique vers le programme appelant, juste après l'instruction d'appel. RET déclenche un transfert des deux octets supérieurs de la pile (où le CALL avait logé l'adresse) dans le PC, ce qui provoque le branchement de retour.

Les instructions de cette catégorie ne font qu'un octet, qui a la structure suivante :

11XXX00Y

Là encore, cette structure est voisine de celle des JUMP et des CALL, avec 00 au lieu de 01 et 10, et de même :

OCTAL	HEX	<B1> MNEMONIC	DESCRIPTION
000	00	NOP	No operation
001	01	LXI B <B2> <B3>	Load immediate into register pair B and C
002	02	STAX B	Store A indirect into M addressed by B and C
003	03	INX B	Increment contents of register pair B and C by 1
004	04	INR B	Increment register B by 1
005	05	DCR B	Decrement register B by 1
006	06	MVI B <B2>	Move immediate into register B
007	07	RLC	Rotate A left
010	08	---	---
011	09	DAD B	Add contents of B,C to H,L and store in H,L
012	0A	LDAX B	Load A indirect from M addressed by B and C
013	0B	DCX B	Decrement contents of register pair B and C by 1
014	0C	INR C	Increment register C by 1
015	0D	DCR C	Decrement register C by 1
016	0E	MVI C <B2>	Move immediate into register C
017	0F	RRC	Rotate A right
020	10	---	---
021	11	LXI D <B2> <B3>	Load immediate into register pair D and E
022	12	STAX D	Store A indirect into M addressed by D and E
023	13	INX D	Increment contents of register pair D and E by 1
024	14	INR D	Increment register D by 1
025	15	DCR D	Decrement register D by 1
026	16	MVI D <B2>	Move immediate into register D
027	17	RAL	Rotate A left through carry
030	18	---	---
031	19	DAD D	Add contents of D,E to H,L and store in H,L
032	1A	LDAX D	Load A indirect from M addressed by D and E
033	1B	DCX D	Decrement contents of register pair D and E by 1
034	1C	INR E	Increment register E by 1
035	1D	DCR E	Decrement register E by 1
036	1E	MVI E <B2>	Move immediate into register E
037	1F	RAR	Rotate A right through carry
040	20	---	---
041	21	LXI H <B2> <B3>	Load immediate into register pair H and L
042	22	SHLD <B2> <B3>	Store L and H into M and M+1, where M = <B2> <B3>
043	23	INX H	Increment contents of register pair H and L by 1
044	24	INR H	Increment register H by 1
045	25	DCR H	Decrement register H by 1
046	26	MVI H <B2>	Move immediate into register H
047	27	DAA	Decimal adjust A
050	28	---	---
051	29	DAD H	Add contents of H,L to H,L and store in H,L
052	2A	LHLD <B2> <B3>	Load L and H with contents of M and M+1, respectively
053	2B	DCX H	Decrement contents of register pair H and L by 1
054	2C	INR L	Increment register L by 1
055	2D	DCR L	Decrement register L by 1
056	2E	MVI L <B2>	Move immediate into register L
057	2F	CMA	Complement A
060	30	---	---
061	31	LXI SP <B2> <B3>	Load immediate into stack pointer
062	32	STA <B2> <B3>	Store A direct into M addressed by <B2> <B3>
063	33	INX SP	Increment register SP by 1
064	34	INR M	Increment contents of M by 1
065	35	DCR M	Decrement contents of M by 1
066	36	MVI M <B2>	Move immediate into M addressed by H and L
067	37	STC	Set carry flip-flop to logic 1
070	38	---	---
071	39	DAD SP	Add stack pointer contents to H,L and store in H,L
072	3A	LDA <B2> <B3>	Load A direct with contents of M addressed by <B2> <B3>
073	3B	DCX SP	Decrement register SP by 1
074	3C	INR A	Increment register A by 1
075	3D	DCR A	Decrement register A by 1
076	3E	MVI A <B2>	Move immediate into register A
077	3F	CMC	Complement carry flip-flop
100	40	MOV B,B	Move contents of register B to register B
101	41	MOV B,C	Move contents of register C to register B
102	42	MOV B,D	Move contents of register D to register B
103	43	MOV B,E	Move contents of register E to register B
104	44	MOV B,H	Move contents of register H to register B
105	45	MOV B,L	Move contents of register L to register B
106	46	MOV B,M	Move contents of M to register B
107	47	MOV B,A	Move contents of register A to register B
110	48	MOV C,B	Move contents of register B to register C
111	49	MOV C,C	Move contents of register C to register C
112	4A	MOV C,D	Move contents of register D to register C
113	4B	MOV C,E	Move contents of register E to register C
114	4C	MOV C,H	Move contents of register H to register C
115	4D	MOV C,L	Move contents of register L to register C
116	4E	MOV C,M	Move contents of M to register C
117	4F	MOV C,A	Move contents of register A to register C
120	50	MOV D,B	Move contents of register B to register D
121	51	MOV D,C	Move contents of register C to register D
122	52	MOV D,D	Move contents of register D to register D
123	53	MOV D,E	Move contents of register E to register D
124	54	MOV D,H	Move contents of register H to register D
125	55	MOV D,L	Move contents of register L to register D
126	56	MOV D,M	Move contents of M to register D
127	57	MOV D,A	Move contents of register A to register D
130	58	MOV E,B	Move contents of register B to register E
131	59	MOV E,C	Move contents of register C to register E
132	5A	MOV E,D	Move contents of register D to register E
133	5B	MOV E,E	Move contents of register E to register E
134	5C	MOV E,H	Move contents of register H to register E
135	5D	MOV E,L	Move contents of register L to register E
136	5E	MOV E,M	Move contents of M to register E
137	5F	MOV E,A	Move contents of register A to register E
140	60	MOV H,B	Move contents of register B to register H
141	61	MOV H,C	Move contents of register C to register H
142	62	MOV H,D	Move contents of register D to register H
143	63	MOV H,E	Move contents of register E to register H
144	64	MOV H,H	Move contents of register H to register H
145	65	MOV H,L	Move contents of register L to register H
146	66	MOV H,M	Move contents of M to register H
147	67	MOV H,A	Move contents of register A to register H
150	68	MOV L,B	Move contents of register B to register L
151	69	MOV L,C	Move contents of register C to register L
152	6A	MOV L,D	Move contents of register D to register L
153	6B	MOV L,E	Move contents of register E to register L
154	6C	MOV L,H	Move contents of register H to register L
155	6D	MOV L,L	Move contents of register L to register L
156	6E	MOV L,M	Move contents of M to register L
157	6F	MOV L,A	Move contents of register A to register L
160	70	MOV M,B	Move contents of register B to M
161	71	MOV M,C	Move contents of register C to M
162	72	MOV M,D	Move contents of register D to M
163	73	MOV M,E	Move contents of register E to M
164	74	MOV M,H	Move contents of register H to M
165	75	MOV M,L	Move contents of register L to M
166	76	HLT	Halt
167	77	MOV M,A	Move contents of register A to M
170	78	MOV A,B	Move contents of register B to register A
171	79	MOV A,C	Move contents of register C to register A
172	7A	MOV A,D	Move contents of register D to register A
173	7B	MOV A,E	Move contents of register E to register A
174	7C	MOV A,H	Move contents of register H to register A
175	7D	MOV A,L	Move contents of register L to register A
176	7E	MOV A,M	Move contents of M to register A
177	7F	MOV A,A	Move contents of register A to register A
200	80	ADD B	Add contents of register B to register A
201	81	ADD C	Add contents of register C to register A
202	82	ADD D	Add contents of register D to register A
203	83	ADD E	Add contents of register E to register A

— Y vaut 1 pour RET, et zéro pour les huit instructions de retour *conditionnel* de sous-programmes.
— XXX vaut

000 pour RNZ	100 pour RPO
001 pour RZ ou RET	101 pour RPE
010 pour RNC	110 pour RP
011 pour RC	111 pour RM

Même remarque ici que celle faite à propos des codes CALL. Par conséquent, cette fois les codes RET s'obtiennent par soustraction de 4 à partir des codes CALL.

CALL *hexpiloc*. Non ! Ne me dites pas encore une fois que l'octal serait mieux pour écrire ces codes ! Je crois que le moment est venu de lui porter le coup de grâce, attention, ça va être un vrai massacre.

Dernièrement, j'étais lancé dans l'agréable passe-temps consistant à dépouiller un programme sur un listing bien compact, en octal, sans le moindre commentaire. En voici la première ligne :

061 000 010 315 265 001 315 335 001 315
175 000 315 175 000 315 142 000 315

et il y en avait pas mal comme ça (je ne suis pas allé au bout, vous verrez bientôt pourquoi).

Première instruction, pas de problème : c'est un LXI SP, ce qui est bien normal comme début, nous l'avons signalé au passage. Son adresse est en deux octets, inversés, quelque part assez loin par conséquent en mémoire pour que la pile grimpe tranquillement.

La seconde instruction est un CALL, avec une adresse plus proche, apparemment 1265 en octal.

<B1> OCTAL	HEX	MNEMONIC	DESCRIPTION
204	84	ADD H	Add contents of register H to register A
205	85	ADD L	Add contents of register L to register A
206	86	ADD M	Add contents of M to register A
207	87	ADD A	Add contents of register A to register A
210	88	ADC B	Add carry and contents of register B to register A
211	89	ADC C	Add carry and contents of register C to register A
212	8A	ADC D	Add carry and contents of register D to register A
213	8B	ADC E	Add carry and contents of register E to register A
214	8C	ADC H	Add carry and contents of register H to register A
215	8D	ADC L	Add carry and contents of register L to register A
216	8E	ADC M	Add carry and contents of M to register A
217	8F	ADC A	Add carry and contents of register A to register A
220	90	SUB B	Subtract contents of register B from register A
221	91	SUB C	Subtract contents of register C from register A
222	92	SUB D	Subtract contents of register D from register A
223	93	SUB E	Subtract contents of register E from register A
224	94	SUB H	Subtract contents of register H from register A
225	95	SUB L	Subtract contents of register L from register A
226	96	SUB M	Subtract contents of M from register A
227	97	SUB A	Clear register A
230	98	SBB B	Subtract carry and contents of register B from A
231	99	SBB C	Subtract carry and contents of register C from A
232	9A	SBB D	Subtract carry and contents of register D from A
233	9B	SBB E	Subtract carry and contents of register E from A
234	9C	SBB H	Subtract carry and contents of register H from A
235	9D	SBB L	Subtract carry and contents of register L from A
236	9E	SBB M	Subtract carry and contents of M from register A
237	9F	SBB A	Subtract carry and contents of register A from A
240	A0	ANA B	AND contents of register B with register A
241	A1	ANA C	AND contents of register C with register A
242	A2	ANA D	AND contents of register D with register A
243	A3	ANA E	AND contents of register E with register A
244	A4	ANA H	AND contents of register H with register A
245	A5	ANA L	AND contents of register L with register A
246	A6	ANA M	AND contents of M with register A
247	A7	ANA A	AND contents of register A with register A
250	A8	XRA B	Exclusive-OR contents of register B with register A
251	A9	XRA C	Exclusive-OR contents of register C with register A
252	AA	XRA D	Exclusive-OR contents of register D with register A
253	AB	XRA E	Exclusive-OR contents of register E with register A
254	AC	XRA H	Exclusive-OR contents of register H with register A
255	AD	XRA L	Exclusive-OR contents of register L with register A
256	AE	XRA M	Exclusive-OR contents of M with register A
257	AF	XRA A	Exclusive-OR contents of register A with register A
260	B0	ORA B	OR contents of register B with register A
261	B1	ORA C	OR contents of register C with register A
262	B2	ORA D	OR contents of register D with register A
263	B3	ORA E	OR contents of register E with register A
264	B4	ORA H	OR contents of register H with register A
265	B5	ORA L	OR contents of register L with register A
266	B6	ORA M	OR contents of M with register A
267	B7	ORA A	OR contents of register A with register A
270	B8	CMP B	Compare contents of register B with register A
271	B9	CMP C	Compare contents of register C with register A
272	BA	CMP D	Compare contents of register D with register A
273	BB	CMP E	Compare contents of register E with register A
274	BC	CMP H	Compare contents of register H with register A
275	BD	CMP L	Compare contents of register L with register A
276	BE	CMP M	Compare contents of M with register A
277	BF	CMP A	Compare contents of register A with register A
300	C0	RNZ	Return from subroutine if zero flip-flop = logic 0
301	C1	POP B	Pop stack and store address in register pair B and C
302	C2	JNZ <B2> <B3>	Jump if zero flip-flop = logic 0
303	C3	JMP <B2> <B3>	Jump unconditionally to M addressed by <B2> <B3>
304	C4	CNZ <B2> <B3>	Call subroutine if zero flip-flop = logic 0
305	C5	PUSH B	Push contents of register pair B and C on stack

<B1> OCTAL	HEX	MNEMONIC	DESCRIPTION
306	C6	ADI <B2>	Add immediate to register A
307	C7	RST 0	Call subroutine at address 000_8
310	C8	RZ	Return from subroutine if zero flip-flop = logic 1
311	C9	RET	Return from subroutine
312	CA	JZ <B2> <B3>	Jump if zero flip-flop = logic 1
313	CB	---	---
314	CC	CZ <B2> <B3>	Call subroutine if zero flip-flop = logic 1
315	CD	CALL <B2> <B3>	Call subroutine located at M = <B2> <B3>
316	CE	ACI <B2>	Add immediate and carry flip-flop to register A
317	CF	RST 1	Call subroutine at address 010_8
320	D0	RNC	Return from subroutine if carry flip-flop = logic 0
321	D1	POP D	Pop stack and store
322	D2	JNC <B2> <B3>	Jump if carry flip-flop = logic 0
323	D3	OUT <B2>	Output to device addressed by <B2>
324	D4	CNC <B2> <B3>	Call subroutine if carry flip-flop = logic 0
325	D5	PUSH D	Push contents of register pair D and E on stack
326	D6	SUI <B2>	Subtract immediate from register A
327	D7	RST 2	Call subroutine at address 020_8
330	D8	RC	Return from subroutine if carry flip-flop = logic 1
331	D9	---	---
332	DA	JC <B2> <B3>	Jump if carry flip-flop = logic 1
333	DB	IN <B2>	Input from device addressed by <B2>
334	DC	CC <B2> <B3>	Call subroutine if carry flip-flop = logic 1
335	DD	---	---
336	DE	SBI <B2>	Subtract immediate and carry flip-flop from register A
337	DF	RST 3	Call subroutine at address 030_8
340	E0	RPO	Return from subroutine if parity flip-flop = logic 0
341	E1	POP H	Pop stack and store address in register pair H and L
342	E2	JPO <B2> <B3>	Jump if parity flip-flop = logic 0
343	E3	XTHL	Exchange top of stack with contents of H and L
344	E4	CPO <B2> <B3>	Call subroutine if parity flip-flop = logic 0
345	E5	PUSH H	Push contents of register pair H and L on stack
346	E6	ANI <B2>	AND immediate with contents of register A
347	E7	RST 4	Call subroutine at address 040_8
350	E8	RPE	Return from subroutine if parity flip-flop = logic 1
351	E9	PCHL	Jump indirect to M addressed by register pair H and L
352	EA	JPE <B2> <B3>	Jump if parity flip-flop = logic 1
353	EB	XCHG	Exchange contents of registers H,L with registers D,E
354	EC	CPE <B2> <B3>	Call subroutine if parity flip-flop = logic 1
355	ED	---	---
356	EE	XRI <B2>	Exclusive-OR immediate with contents of register A
357	EF	RST 5	Call subroutine at address 050_8
360	F0	RP	Return from subroutine if sign flip-flop = logic 0
361	F1	POP PSW	Pop stack and store in register A and flag flip-flops
362	F2	JP <B2> <B3>	Jump if sign flip-flop = logic 0 [positive sign]
363	F3	DI	Disable interrupt
364	F4	CP <B2> <B3>	Call subroutine if sign flip-flop = logic 0
365	F5	PUSH PSW	Push contents of register A and flags on stack
366	F6	ORI <B2>	OR immediate with contents of register A
367	F7	RST 6	Call subroutine at address 060_8
370	F8	RM	Return from subroutine if sign flip-flop = logic 1
371	F9	SPHL	Transfer contents of registers H,L to stack pointer
372	FA	JM <B2> <B3>	Jump if sign flip-flop = logic 1 [minus sign]
373	FB	EI	Enable interrupt
374	FC	CM <B2> <B3>	Call subroutine if sign flip-flop = logic 1
375	FD	---	---
376	FE	CPI <B2>	Compare immediate with contents of register A
377	FF	RST 7	Call subroutine at address 070_8

Or, à cette adresse, rien de bien valable : on tombe en plein milieu d'un JUMP, sur son deuxième octet... Qu'y a-t-il donc ? Regardons les choses d'un peu plus près :

— notre 265 octal donne l'octet :
10 110 101 en binaire

— notre 001 octal donne l'octet :
00 000 001 en binaire

Inversons les deux octets pour avoir l'adresse normale :

000000110110101

Séparons maintenant en tranches de 3 pour revenir à l'octal :

...110 110 101, soit 665 !!

Horreur ! Il n'est pas possible de permuter les deux nombres de trois chiffres en octal pour avoir directement l'adresse... Il faut faire toute une gymnastique (même s'il n'y a pas à permuter, le problème reste entier...).

Après ça, les partisans de l'octal qui riaient au passage (débile, c'est vrai) sur les œufs et le reste, leur rire a jauni. Le seul avantage restant à l'octal nous semble être au niveau matériel : réalisation d'un mini-clavier/visu moins cher qu'en hexa.

Reprenons le listing comme on l'aurait sorti en hexa (à propos, c'est un programme qui visualise le jeu de la vie de John Conway sur une TV couleur : bleu pour la vie, vert pour une naissance, rouge pour la mort, vous l'aurez sur Alcyane avec l'interface graphique TV couleur) :

31 00 02 CD 5B 01, etc...

MOVES

(vers) \ (de)	A	B	C	D	E	H	L	M	I	LDA
A	7F	78	79	7A	7B	7C	7D	7E	3E	3A
B	47	40	41	42	43	44	45	46	06	
C	4F	48	49	4A	4B	4C	4D	4E	0E	
D	57	50	51	52	53	54	55	56	16	
E	5F	58	59	5A	5B	5C	5D	5E	1E	
H	67	60	61	62	63	64	65	66	26	
L	6F	68	69	6A	6B	6C	6D	6E	2E	
M	77	70	71	72	73	74	75		36	
STA	32									

ARITHMETIC

	A	B	C	D	E	H	L	M	I
ADD	87	80	81	82	83	84	85	86	C6
ADC	8F	88	89	8A	8B	8C	8D	8E	CE
SUB	97	90	91	92	93	94	95	96	D6
SBB	9F	98	99	9A	9B	9C	9D	9E	DE
ANA	A7	A0	A1	A2	A3	A4	A5	A6	E6
XRA	AF	A8	A9	AA	AB	AC	AD	AE	EE
ORA	B7	B0	B1	B2	B3	B4	B5	B6	F6
CMP	BF	B8	B9	BA	BB	BC	BD	BE	FE
INR	3C	04	0C	14	1C	24	2C	34	
DCR	3D	05	0D	15	1D	25	2D	35	

PAIRES

	B	D	H	SP	PSW
LXI	01	11	21	31	
DAD	09	19	29	39	
LDAX	0A	1A	(7E)		
STAX	02	12	(77)		
INX	03	13	23	33	
DCX	0B	1B	2B	3B	
PUSH	C5	D5	E5		F5
POP	C1	D1	E1		F1
LHLD			2A		
SHLD			22		

DIVERS

échanges HL	XCHG EB	XTHL E3	SPHL F9	PCHL E9				
accumulateur	DAA 27	CMA 2F	STC 37	CMC 3F				
rotations	RLC 07	RRC 0F	RAL 17	RAR 1F				
commande	NOP 00	HLT 76	DI F3	EI FB				
entrée/sortie	OUT D3	IN DB						
restart	0 C7	1 CF	2 D7	3 DF	4 E7	5 EF	6 F7	7 FF

SAUTS

		Z	C	P	S	$\overline{Z}$	$\overline{C}$	$\overline{P}$	$\overline{S}$
JMP	C3	CA	DA	EA	FA	C2	D2	E2	F2
CALL	CD	CC	DC	EC	FC	C4	D4	E4	F4
RET	C9	C8	D8	E8	F8	C0	D0	E0	F0

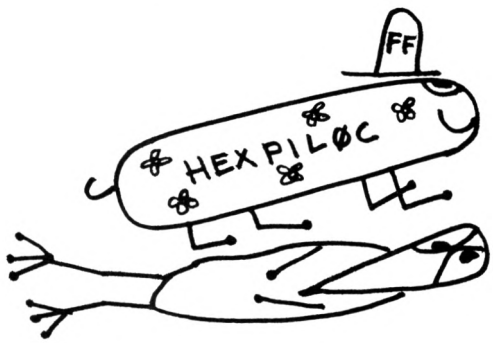
NON UTILISÉ

00XXX000 (XXX≠000)
11XX1101
11001011
11011001

d'où avec les adresses normales en hexa :

LXI SP 0200, CALL 015B, etc...

Pas de commentaires, le triomphe doit être modeste. Pour permettre aux pécheurs repentis



de revenir dans le droit chemin, mais aussi aux hérétiques d'aller vers l'octal, nous avons reproduit un listing donnant :

octal/hexa/mnémonique/description de l'instruction

Ce listing permet en outre de se rafraîchir rapidement la mémoire sur la signification de LDAXB ou DAD H. Il est en anglais, mais c'est un vocabulaire limité qui, arrivé jusqu'ici, est sans doute bien maîtrisé, même par les anglophobes. Pour chercher un libellé, vous pouvez consulter le classement par groupes donné précédemment, pour trouver l'instruction, et entrer ensuite par son code hexa.

Enfin, dernier service, vous avez sous la main une conversion octal-hexa de 0 à 377 ou FF. **RET**.

Nous voici revenus au programme principal, c'est le moment d'enchaîner. Aucun drapeau n'est affecté par ces instructions. Celles qui sont conditionnelles effectuent un retour si la condition spécifiée est vraie. Sinon, on continue en séquence.

RET *Return* (retour)
11001001 soit **C9**
Le retour s'effectue toujours (retour inconditionnel).

RC *Return if Carry* (retour si report)
11011000 soit **D8**
Le retour s'effectue si le flag CY est à un.

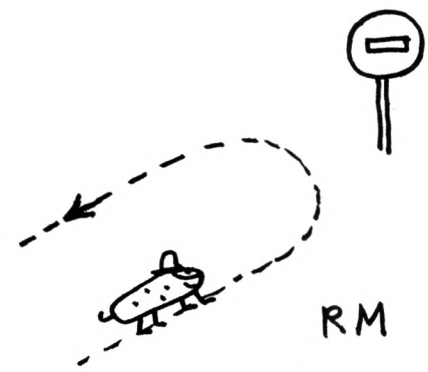
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LXI B	STAX B	INX B	INR B	DCR B	MVI B	RLC	-	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C	RRC
1	-	LXI D	STAX D	INX D	INR D	DCR D	MVI D	RAL	-	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E	RAR
2	-	LXI H	SHLD	INX H	INR H	DCR H	MVI H	DAA	-	DAD H	LHLD	DCX H	INR L	DCR L	MVI L	CMA
3	-	LXI SP	STA	INX SP	INR M	DCR M	MVI M	STC	-	DAD SP	LDA	DCX SP	INR A	DCR A	MVI A	CMC
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ	JMP	CNZ	PUSH B	ADI	RST 0	RZ	RET	JZ	-	CZ	CALL	ACI	RST 1
D	RNC	POP D	JNC	OUT	CNC	PUSH D	SUI	RST 2	RC	-	JC	IN	CC	-	SBI	RST 3
E	RPO	POP H	JPO	XTHL	CPO	PUSH H	ANI	RST 4	RPE	PCHL	JPE	XCHG	CPE	-	XRI	RST 5
F	RP	POP PSW	JP	DI	CP	PUSH PSW	ORI	RST 6	RM	SPHL	JM	EI	CM	-	CPI	RST 7

RNC *Return if No Carry* (retour si non report)
11010000 soit **D0**
Le retour s'effectue si le flag CY est à zéro.

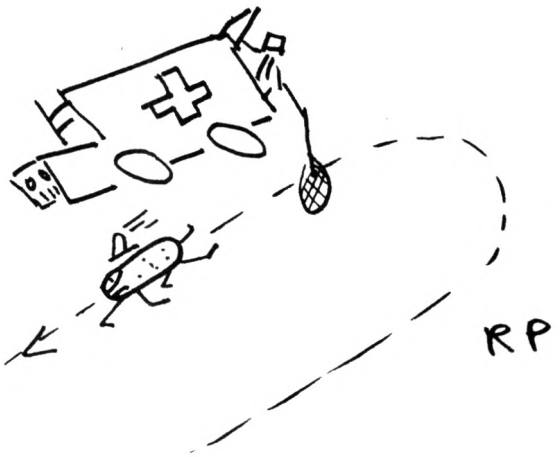
RZ *Return if Zero* (retour si zéro)
11001000 soit **C8**
Le retour s'effectue si le contenu de l'accu est égal à zéro à la suite d'une opération (drapeau Z égal à un, voir remarque pour le call conditionnel CZ).

RNZ *Return if Not Zero* (retour si non zéro)
11000000 soit **C0**.
Le retour s'effectue si le contenu de l'accu est non nul à la suite d'une opération (drapeau Z égal à zéro, voir instruction précédente).

RM *Return if Minus* (retour si moins)
11111000 soit **F8**
Le retour s'effectue si le contenu de l'accu est négatif à la suite d'une opération (flag S égal à un).



RP *Return if Plus* (retour si plus)
11110000 soit **F0**
Le retour s'effectue si le contenu de l'accu est positif à la suite d'une opération (flag S égal à zéro).



RPE *Return if Parity Even* (retour si parité paire) **11101000** soit **E8**

Le retour s'effectue si l'accu à la suite d'une opération contient un nombre pair de bits un (flag P égal à un).

RPO *Return if Parity Odd* (retour si parité impaire) **11100000** soit **E0**

Le retour s'effectue si l'accu à la suite d'une opération contient un nombre impair de bits un (drapeau P égal à zéro).

3 - 8 - 6. Exemples de programmation (structurée)

L'utilisation de sous-programmes est très commode en programmation. Rappelons brièvement de quoi il s'agit, et en particulier la différence entre macro-instructions et sous-programmes.

Lorsqu'on rédige des programmes, on se rend vite compte qu'on est régulièrement amené à utiliser de façon répétitive des séquences identiques d'instructions accomplissant un travail bien défini. Donnons-en des exemples :

— lecture d'un caractère du clavier,
— visualisation d'un caractère, ou d'une image entière,
— impression d'un caractère, ou d'une ligne entière,
— avance du papier,
— écriture d'un caractère sur cassette audio, ou d'un bloc,
— lecture idem,
— conversions de codes,
— opérations arithmétiques décimales en virgule fixe,
— idem virgule flottante, etc., etc.

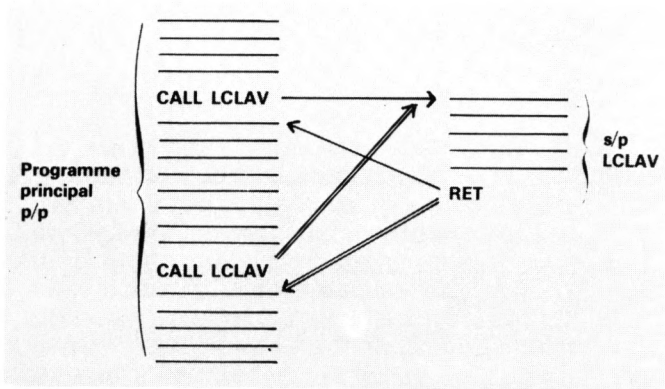
Quand on se sert du macro-assembleur 8K d'Alcyane, on peut donner des noms symboliques à de telles séquences d'instructions, par exemple LCLAV à la première, en écrivant LCLAV dans la colonne label, et MACRO, sur la même ligne, dans la colonne code opération. On met à la suite toutes les instructions de la séquence, et on la termine par ENDM (fin macro) dans la colonne code.

Ensuite, dans le reste du programme, le fait d'écrire simplement LCLAV dans la colonne des codes opérations provoquera l'appel de la séquence d'instruction qui viendra purement et simplement remplacer LCLAV au fil du texte.

On peut se constituer progressivement une bibliothèque de macros, et à la limite on crée ainsi facilement des langages spécialisés, puissants et efficaces (21).

Chaque fois donc qu'une macro est invoquée, il y a insertion de la séquence. C'est pratique, mais dans la mesure où ces appels sont répétés, l'encombrement du programme croît rapidement (jadis, on appelait les macros *sous-programmes ouverts*).

La solution préférable est alors d'employer des sous-programmes (jadis appelés *sous-programmes fermés*) au lieu des macros. Dans ce cas, les séquences ne figureront qu'une seule fois en mémoire, et on ira les parcourir en quittant le programme principal :



Pour que ce schéma fonctionne, plusieurs problèmes sont à résoudre :

— le retour. Comme le s/p peut être utilisé à partir de nombreux points de départ, il faut connaître la valeur du PC au moment du branchement, et la ré-utiliser pour revenir : c'est le rôle du couple CALL/RET et de la pile d'adresses à gestion automatique du 8080,
— le passage des paramètres. En effet, si l'on utilise un s/p, c'est pour obtenir un résultat, à partir de données ou non. Avec LCLAV, pas de données à communiquer par le p/p au s/p. Ce

(21) Voir l'article «Redécouverte des macro-instructions», par Pierre KRAUS, *L'Informatique nouvelle* n° 48 Février 1974.

dernier retourne un résultat (le caractère lu) tout simplement en le mettant dans l'accu : c'est le moyen le plus universellement employé.

Dans le cas où le p/p doit passer des données au s/p, le plus efficace est de les loger dans des registres généraux, juste avant le CALL : on simplifie ainsi nettement la programmation, et on gagne du temps.

A son tour, un s/p peut faire appel à un autre s/p, et ainsi de suite, à tiroirs : aucun problème avec les CALL s/p, les valeurs successives de PC dans la pile jouant le rôle des cailloux du Petit Poucet. La seule limite est la hauteur permise de la pile, qui croît de deux octets à chaque appel (et diminue de deux octets à chaque retour). Elle ne dépend donc que de l'espace prévu en mémoire vive à cet usage, le nombre de niveaux de s/p imbriqués étant donc illimité en pratique.

Au-delà des avantages que nous venons de voir, les s/p présentent un intérêt plus subtil : celui d'améliorer nettement la qualité de la conception des programmes. On parle beaucoup de programmation structurée, et c'est sûr que l'abondance des CALL et RET d'Alcyane va nous conduire sans douleur à la fameuse approche *top-down*.

Pour le détail, lire les nombreux articles de Marie-Thérèse Bertini et Yves Tallineau sur le sujet (*L'informatique nouvelle*, depuis fin 1972, *passim,* et surtout «le Cobol Structuré» et «le Laboratoire Informatique») et l'article d'Alain Leconte, dérivé de leurs travaux, «L'Assembleur et la programmation structurée» (ibid., n° 57, décembre 1974).

Pour une idée générale et affaiblie de la question, lire simplement les quelques phrases qui suivent.

Lorsqu'on retourne dans sa tête le projet du futur programme, on en voit assez rapidement l'architecture globale. Dès cet instant, commencez à l'écrire *sans rentrer dans le détail*, en utilisant une série de CALL à des s/p imaginaires, que vous entrevoyez approximativement mais que vous n'écrivez pas pour le moment. Cette séquence sera l'épine dorsale de votre p/p, le module de commande, la *structure* si vous préférez : les CALL seront en effet souvent conditionnels, mélangés à des comparaisons, des tests, etc.

Ensuite, écrivez les s/p invoqués *en employant la même approche* : ils se décomposeront alors eux-mêmes en s/p. Recommencez à nouveau le processus, et au bout de quelques rounds vous aboutirez à des s/p de quelques dizaines d'instructions, que vous écrirez alors en totalité très simplement. Cette *approche descendante* du général au particulier, c'est le *top-down*, par opposition au *bottom-up, approche montante* où l'on grappille des détails dans le bas dès le début. Il ne reste plus alors qu'à organiser l'ensemble par niveaux en mettant à la suite dans le listing :

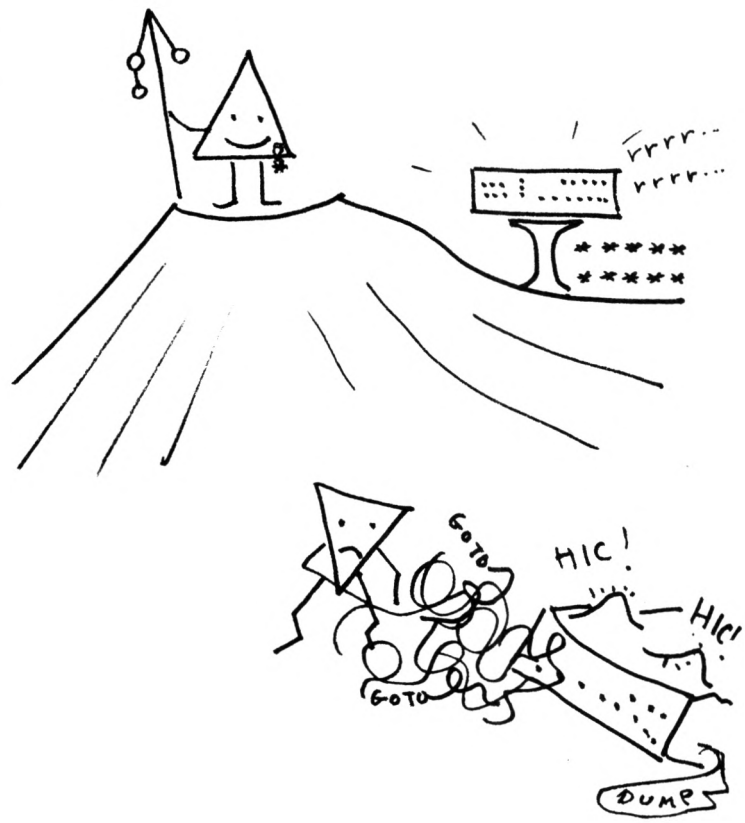
— la structure
**********NIVEAU 1

— les s/p du premier niveau
**********NIVEAU 2

— les s/p du deuxième niveau
**********NIVEAU 3

etc.

On est loin de la justification des s/p par la répétitivité de leur utilisation et l'économie de place ! Ici, un s/p peut très bien n'être appelé qu'une seule fois, sa justification est le désencombrement du p/p qui permet de dégager la structure.



Pour finir, la bande dessinée du n° de janvier 1976 de la revue citée : l'astérisque, que nous vous conseillons d'utiliser pour séparer par des lignes les différents niveaux, y raconte ses avatars avec MTB et YT. Vous serez alors presque devenu un *top-downa* bien structuré, quittant votre état primitif de *bottomupa*, spécialiste de la fabrication de programmes en spaghettis échevelés.

3 - 8 - 7. Instructions relatives aux interruptions

Il y en a quatre : un saut spécial vers s/p, les instructions permettant d'autoriser ou non les interruptions, et une halte en attente d'interruption.

RST *Restart* (redémarrage)
11AAA111 soit **C7, CF, D7, DF, E7, EF, F7, FF.**
AAA peut prendre les 8 valeurs binaires de 000 à 111. L'instruction RST met sur la pile le PC, préparant ainsi le retour par une instruction de la catégorie RET, et provoque un branchement à l'adresse :

0000000000AAA000

Cette instruction d'un octet seulement permet donc de réaliser toutes les fonctions d'un CALL, mais avec seulement huit adresses possibles de branchement, échelonnées de 8 en 8 octets : 0, 8, 16, 24, 32, 40, 48 et 56, en décimal (on voit bien que AAA est multiplié par 8, par trois décalages à gauche).

Nous entrons ici dans un domaine où le hardware intervient au milieu du software. Rappelons qu'une interruption a pour effet de suspendre le déroulement normal du programme, à la suite d'une intervention étrangère au dit programme. Les interruptions sont de deux classes :

— les interruptions *internes*. Elles sont déclenchées par le processeur, à la suite d'un événement anormal : division par zéro, erreur détectée du bloc de calcul, etc. Cette classe d'interruptions n'existe pas en général dans les μP, et il n'y en a pas dans le 8080 (comme il n'y a pas d'instruction de division... c'est au s/p de prendre ses précautions en vérifiant que le diviseur n'est pas nul, avant de commencer à mouliner dans le vide *in eternam* ; et si l'on veut vérifier les calculs, on les fait deux fois, en clair et en complément, puis on complémente l'un des résultats et on compare : mais c'est inutile en pratique, on n'est plus à l'âge des tubes à vide...),
— les interruptions *externes*. Comme leur nom l'indique, elles proviennent de l'extérieur du processeur : de capteurs de mesure ou de détection d'états dans un système de commande de processus, de périphériques signalant la fin d'une opération de lecture ou d'écriture, etc.

C'est par la ligne INT qu'arrive le signal commandant une interruption sur le 8080.

Comme on le voit en examinant le diagramme d'états, le programme ne peut être interrompu que si la bascule INTE est armée (en position un, démasquant ainsi les interruptions : le positionnement de la bascule est réalisé par les instructions EI et DI). Si c'est le cas, et qu'un signal d'interruption est reçu, lorsqu'on arrive à la fin de l'instruction en cours, l'incrémentation de PC a déjà eu lieu et une série d'événements se produit :

— la bascule INTE est désarmée (INTE = 0), de façon à ce qu'une seconde interruption qui viendrait à se manifester ne puisse pas tout perturber immédiatement,
— le hardware, sous l'action du signal en provenance de la source de l'interruption, entonne de force dans le μP une instruction RST en l'envoyant dans le bus données. Entendons nous bien : cette instruction n'existe pas dans la mémoire, et n'est donc pas lue par le μP. Dans un système utilisé par exemple pour du contrôle de processus industriel, on aura concocté toute une logique en TTL qui recevra jusqu'à 8 lignes d'entrée, reliées chacunes à de multiples sources d'interruption. Chaque ligne provoquera un RSTn, n de 0 à 7, un chiffre différent par ligne, et la logique TTL envoie 8 bits 11XXX11 sur le bus data en même temps qu'un signal sur INT. On dira alors qu'on a *huit niveaux d'interruptions vectorisés hardware*. En effet, aucune intervention de soft pour différencier les huit classes de sources d'interruptions, classes correspondant en général à des priorités, des urgences plus ou moins grandes de traitement : chacune d'elles ira atterrir sur un des 8 s/p logés en tête de mémoire (ou plus exactement un des 8 JUMP branchant sur les s/p placés ailleurs). Les s/p identifieront alors en fonction d'autres signaux *la* source qui a produit l'interruption, et exécuteront l'action appropriée (lire une pression dans un registre, arrêter un ventilateur, ou appeler les pompiers...).

  Dans la version de base d'Alcyane, un seul niveau d'interruption existe en standard : c'est le RST 7 qui, si l'on actionne par exemple l'interrupteur INT au pupitre, provoquera un branchement en 38 H (aïe, pas beau en hexa, en octal on aurait 70 : pas de doute, Federico Faggin et ses collègues travaillaient en octal... les 8 RST donnent alors 0, 10, 20, 30, 40, 50, 60 et 70 comme les malins l'avaient déjà repéré sur la structure de l'adresse de branchement de RST). La logique qui fabrique l'octet FF du RST 7 est fort commodément contenue dans le DIP 8228, déjà relié aux huit lignes D_0 à D_7 puisqu'entre autres il sert de buffer sur le bus données, et reçoit le mot d'état du μP en T_2 comme vu,
— la progression de PC qui se produit normalement pendant l'exécution d'une instruction est ici suspendue,
— PC est empilé, en l'état qu'il avait donc à la fin de l'instruction i précédant l'instruction de

RST : par conséquent, lorsqu'on reviendra par RET après exécution du s/p de traitement de l'interruption, ce sera à l'octet suivant l'instruction i.

On voit donc que RST joue le rôle d'un CALL déclenché de l'extérieur, mais d'un CALL très compact puisqu'il ne fait qu'un octet de long au lieu de trois.

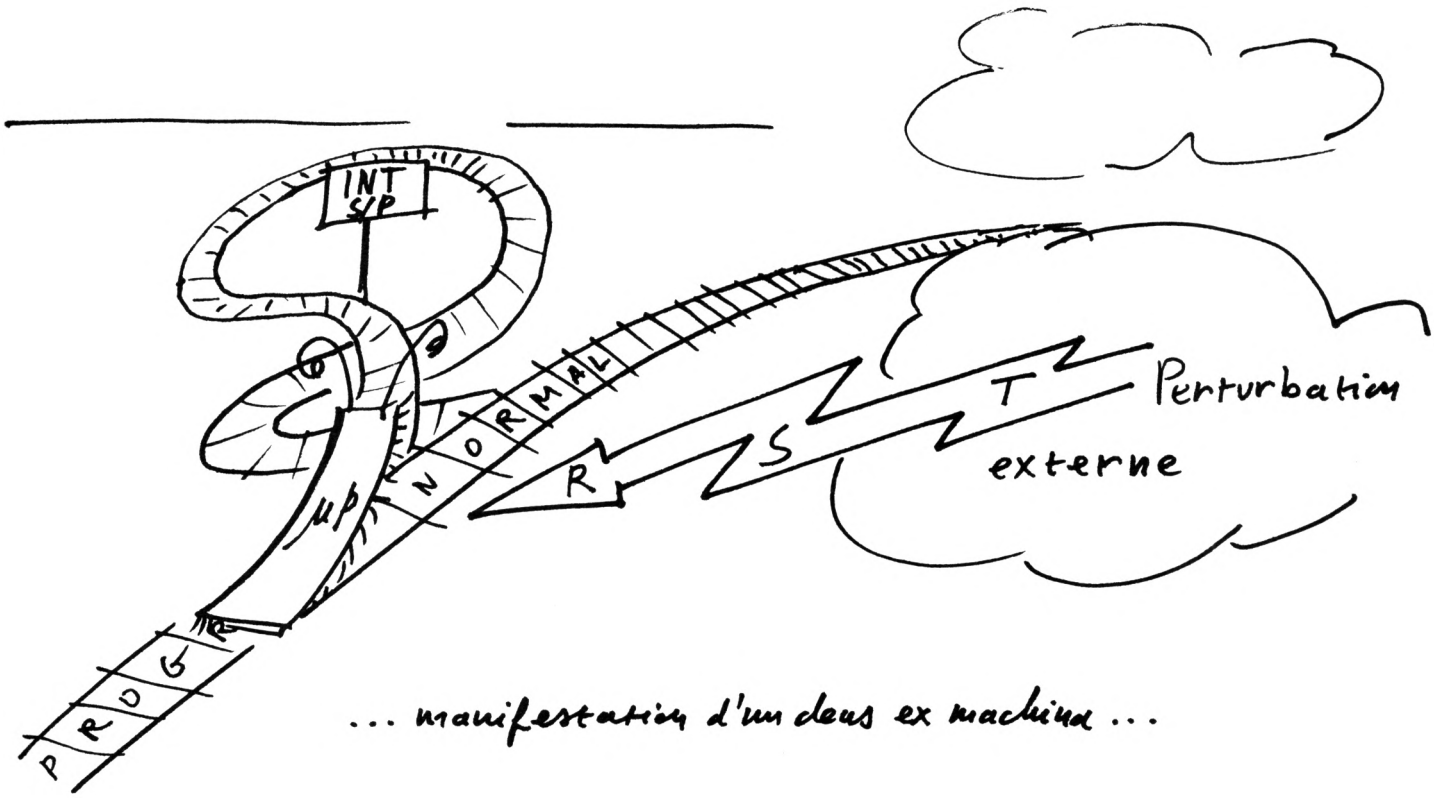
Tel que décrit précédemment, RST est un *deus ex machina* n'existant pas dans le programme, et qui arrivant du dehors sur ses huit fils propulse le CPU hors de ses rails. Mais RST peut aussi être utilisé tout bêtement comme instruction dans un programme, pour appeler des s/p très fréquemment utilisés dont on aura logé les premières instructions aux huit emplacements prévus (ou plutôt aux sept, en laissant à RST 7 le sien) : on économise ainsi deux octets à chaque appel des s/p en question, ce qui n'est pas négligeable.

Remarquons que RST 7 s'écrivant FF en langage machine, toute zone mémoire remplie de bits 1 est une succession de RST 7. Plutôt donc que de remettre la mémoire à zéro avant de charger un programme (succession de NOP), il vaut mieux la remettre à FF, et si une erreur de branchement nous fait atterrir dans le no man's land, on ira systématiquement en 56 où on pourra mettre un JUMP 38 H qui nous bloquera net. Il sera alors facile au pupitre de récupérer *la fausse adresse de branchement*.

De façon à éviter de perturber le programme, en écrivant un s/p d'interruption on devra faire attention à plusieurs points :

— supposons par exemple que l'interruption se produise juste avant une instruction JC, ou JZ : si le s/p exécute des instructions qui modifient le contenu de l'accu ou le drapeau CY, au retour le programme donnera un résultat faux. Il faut donc dans ce cas sauvegarder au moins les 5 drapeaux et l'accu, c'est-à-dire le PSW, en l'empilant par PUSH PSW. Remarquons que cette précaution peut déjà être nécessaire dans le cas d'un CALL s/p, mais là on sait vers quel s/p on va, puisqu'on l'appelle soi-même, et on sait exactement les précautions à prendre le cas échéant. Ici, par contre, une interruption se manifeste brutalement en n'importe quel point du programme, et elle déclenchera l'exécution d'un des s/p sans qu'on sache lequel à l'avance. Par dessus le marché, il faut tenir compte que :

— en général, lorsqu'on débute le s/p d'interruption, dès qu'on a sauvegardé le PSW, il faut réarmer le flip-flop INTE (pour ne pas risquer de laisser passer une autre interruption) à l'aide de l'instruction EI. Dès lors, on peut quitter à tout instant le s/p pour passer dans un autre, et ainsi de suite...



Le s/p d'interruption s'écrira donc en général :

```
PUSH    PSW
EI
•
•       Instructions de
•       service de
        l'interruption
POP     PSW
RET
```

Voyons maintenant les trois autres instructions relatives aux interruptions.

EI *Enable Interrupts* (autoriser les interruptions)
11111011 soit **FB.**

Cette instruction met à 1 la bascule INTE, permettant au μP de prendre en compte les interruptions. Aucun drapeau n'est affecté.

DI *Disable Interrupts* (empêcher les interruptions)
11110011 soit **F3.**

Cette instruction met à 0 la bascule INTE, empêchant le μP de prendre en compte les interruptions. Aucun drapeau n'est affecté.

HLT *Halt* (halte)
01110110 soit **76.**

Le PC est incrémenté jusqu'à prendre la valeur de l'adresse de l'octet qui suit l'instruction HLT. Le μP stoppe alors complètement son fonctionnement en entrant dans l'état WAIT comme nous l'avons vu dans l'analyse des cycles : seule une interruption pourra le sortir de sa léthargie... et pour cela, non seulement il faut qu'il puisse en arriver une par une ligne installée, mais encore que INTE soit à l'état 1. Si INTE = 0 et qu'on exécute un HLT, la situation paraît sans espoir ! On peut heureusement éviter d'avoir recours aux grands moyens (coupure et rétablissement de l'alimentation électrique du μP) en actionnant la clef RAZ du pupitre d'Alcyane : elle remet à zéro le PC, les registres tampons des unités périphériques et nous ramène au départ.

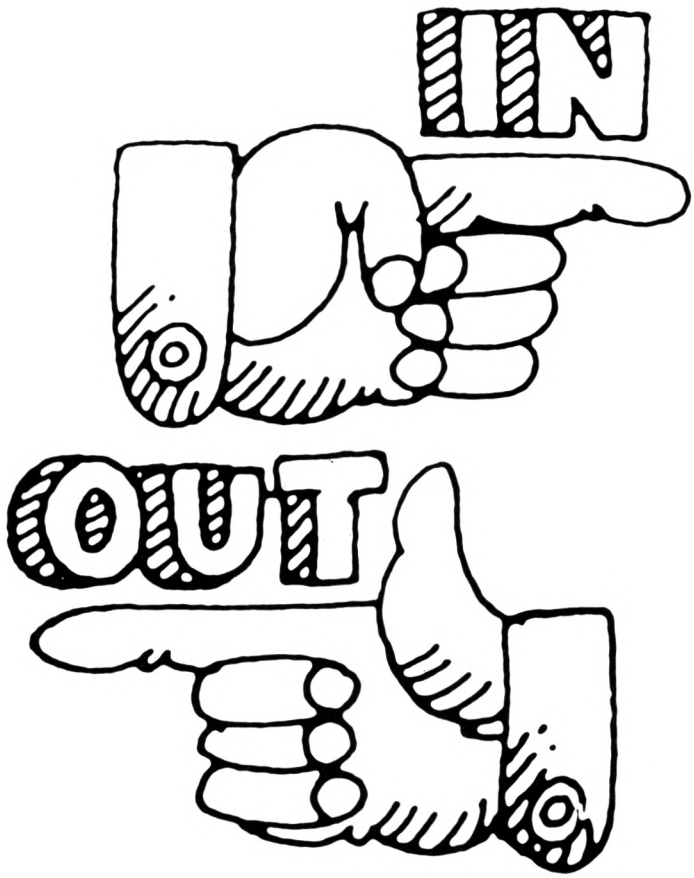
Retenons-en une leçon, il vaut mieux ne pas utiliser l'instruction HALT pour stopper, mais comme nous l'avons déjà signalé un JUMP sur lui-même d'où en sortira par divers moyens sans problème pour redémarrer le programme.

3 - 8 - 8. Instructions d'entrée et de sortie

Ces instructions, au nombre de deux, ont chacune deux octets de longueur. Le premier octet est le code opération, identique à un bit près pour les deux instructions : ce bit, le quatrième en partant de la droite, vaut 1 pour IN et 0 pour OUT, ce qui est facile à se rappeler.

Le deuxième octet contient l'adresse du périphérique concerné, en huit bits par conséquent. Cette adresse a une structure qui dépend du hardware utilisé, de la façon dont les interfaces d'entrée/sortie en particulier ont été conçus. Dans une machine de contrôle de processus industriel où l'on désire souvent connecter un maximum de lignes, l'adresse sera en général un nombre binaire allant par conséquent de 0 à 255, et permettant d'adresser 256 périphériques en entrée et 256 autres en sortie.

Pour des applications informatiques style gestion et calcul, on peut adopter d'autres schémas plus parlants, et c'est ce qui a été fait pour Alcyane (en utilisant en particulier le Décimal Codé en Binaire à 4 bits).



IN *Input* (entrée, entrer)
11011011 soit **DB**, suivi de l'octet adresse du périphérique.

Un octet est lu sur le périphérique spécifié, et placé dans l'accumulateur (dont le contenu précédent est détruit). Aucun drapeau de condition n'est affecté.

Remarquons que la valeur hexa du code opération est l'abréviation de... Data Bus ! On se deman-

de si c'est bien le hasard, qui fait que l'instruction se lit dans la liste sour la forme

```
DB          IN
```

entrée sur le bus des données, on ne saurait être plus clair.

OUT *Output* (sortie, sortir)
11010011 soit **D3**, suivi de l'octet adresse du périphérique.

Le contenu de l'accumulateur est envoyé vers le périphérique spécifié (accu inchangé). Aucun flag de condition n'est affecté.

On voit donc que les échanges d'information avec l'extérieur se font octet par octet sous contrôle du programme (ce n'est pas la seule méthode, on a dit un mot du DMA à propos de HOLD). Sans DMA donc, le rythme le plus rapide de sortie sera obtenu en écrivant

```
LDA          RESUL
OUT          PERIF
LDA          RESUL + 1
OUT          PERIF
LDA          RESUL + 2
OUT          PERIF
             etc.
```

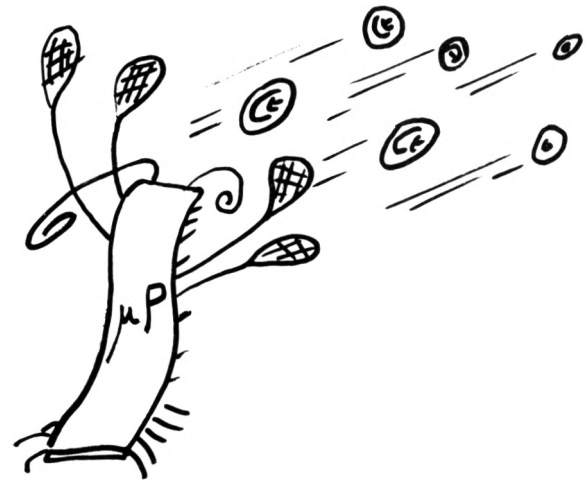
LDA demande 13 états successifs, et OUT 10, soit 23 par octet. A 0,5 microseconde par état, cela fait 11,5 microsecondes par octet : le rythme de sortie est donc de 86.956 octets/seconde. On est loin du million et plus du DMA (pour le floppy par exemple), mais dans un domaine encore respectable.

Avec une séquence moins gaspilleuse de place, du style

```
                                    Etats
ECRSY   MOV     A, M                  7
        OUT     PERIF                10
        INX     H                     5
        DCR     C                     5
        JNZ     ECRSY                10
        •
        •                            37
```

on sort à 18,5 microseconde/octet, soit 54 054 octets/seconde.

Si enfin il faut tester l'état du périphérique avant de lui envoyer un nouvel octet, on écrira par exemple




```
                                    Etats
ECRAS   IN      ETPER                10
        JZ      ECRAS                10
        MOV     A, M                  7
        OUT     PERIF                10
        INX     H                     5
        DCR     C                     5
        JNZ     ECRAS                10
        •
        •                            47 au plus
```

Selon le nombre de tours de boucle dans les deux premières instructions, on aura des débits variables de l'ordre de quelques dizaines de kilo-octets/seconde, ce qui reste bien raisonnable pour les périphériques courants.

3 - 9. DU COTE DE LA PROGRAMMATION

Et voilà ! Nous en avons fini avec les instructions du 8080. On peut maintenant passer à un autre style d'exercices (que nous venons d'ailleurs d'aborder avec notre étude de vitesse d'entrée/sortie) : un peu de programmation avec quelques calculs de temps pour fixer les ordres de grandeur des possibilités d'Alcyane, puis une réflexion sur programmation/software/micro-ordinateurs.

3 - 9 - 1. Petite programmation pondérée

Traiter des quantités isolées de 8 à 16 bits est un travail trop facile pour un µP à mot de 8 ou 16 bits, il est par trop avantagé. Aussi allons-nous soumettre le 8080 à un vrai test correspondant à la vie réelle : le travail sur chaînes de caractères.

Deux nombres décimaux de 16 chiffres chacun sont stockés dans la mémoire, sous forme par conséquent de chaînes de 8 octets. Supposons que les octets de poids croissant correspondent à des adresses croissantes, et que les adresses des octets de début de chaque donnée soient DDECA et DDECB. Nous voulons réaliser l'opération :

Donnée A + Donnée B, résultat dans l'emplacement de Donnée B

le programme correspondant en assembleur sera le suivant :

	Code opération	Opérande	Octets	Etats
	LXI	H, DDECA	3	10
	LXI	D, DDECB	3	10
	MVI	C,8	2	7
	XRA	A	1	4
ADCH :	LDAX	D	1	7
	ADC	M	1	7
	DAA		1	4
	STAX	D	1	7
	INX	H	1	5
	INX	D	1	5
	DCR	C	1	5
	JNZ	ADCH	3	10
	•			
	•			

Les deux premières instructions chargent les adresses de début des chaînes dans HL et DE, qui sont alors transformés en pointeurs de 16 bits chacun, pointant dans la bonne direction dans le champ 64K. Le MVI charge dans C la constante qui nous permettra de faire 8 tours de boucle pour additionner nos 16 chiffres décimaux. XRA A est utilisée comme déjà vu pour remettre à zéro à la fois l'accu *et* le bit de report CY.

Remarquons au passage que les diverses classes d'instructions dont nous disposons nous permettent tout un jeu de combinaisons dans les remises à zéro de l'accu et de CY :

— XRA	A	remet les deux à zéro,
— ANI	FF	remet CY à zéro sans changer le contenu de l'accu
— XRI	0	produit le même effet,
— ORA	A	de mmême,
— MVI	A,0	remet l'accu à zéro sans changer CY etc., il y en a d'autres.

Tout est prêt maintenant pour l'addition des chaînes, on entre dans la boucle. Chargement du premier octet du deuxième nombre, addition avec le premier octet du premier nombre *et le bit de report CY*, nul au premier coup. Transformation en pur décimal du résultat de l'addition par DAA, avec éventuelle émission vers CY d'un report qui sera pris en compte au tour suivant. Stockage des deux chiffres décimaux résultant dans le premier octet du deuxième nombre. Progression des deux pointeurs HL et DE d'une unité. Décomptage d'une unité de la constante chargée dans C. Test de la valeur résiduelle : si elle n'est pas encore nulle, revenir au LDAX pour l'octet suivant.

On remarque qu'avec la méthode de manipulation des pointeurs, il n'y a aucun problème pour manier les chaînes dans un sens, dans l'autre, en sens contraire, un octet sur 2, ou 3, etc., c'est la souplesse totale : ceci est particulièrement pratique pour

— les transferts de chaînes d'un point sur l'autre
— les condensations/décondensations du numérique
— les s/p de compression/décompression
— l'examen des chaînes pour transcodage
etc.

Voyons maintenant le temps de traitement. L'initialisation, qui s'effectue une fois, représente 31 états, soit 15,5 microsecondes. La boucle comporte 50 états, soit 25 microsecondes par tour. Pour 16 chiffres on la parcourt 8 fois, ce qui donne 15,5 + 8 X 25 = 215,5 microsecondes.

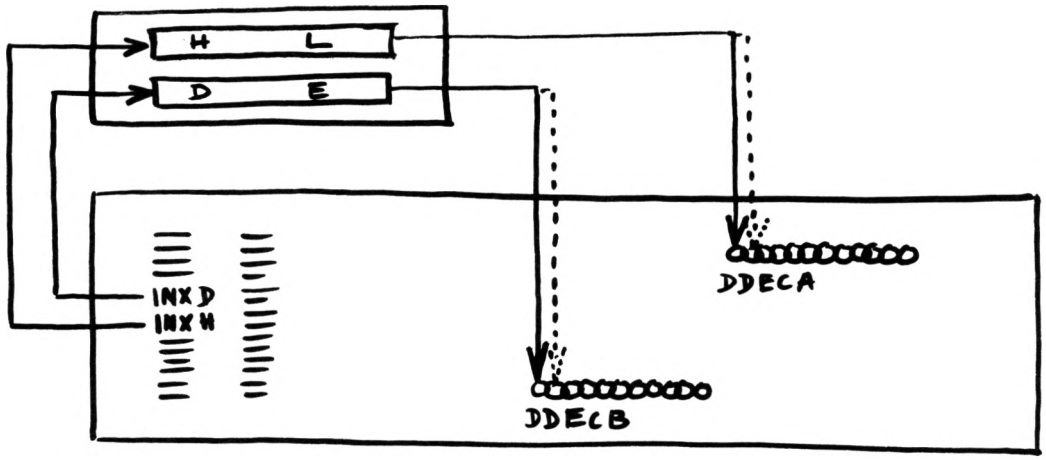
Comment juger ce temps ?

Le mieux, bien sûr, est de revenir aux sources en prenant comme étalons de comparaison les machines les plus répandues, dont la puissance de traitement nous «parle» directement : les deux chevaux de bataille d'IBM, le 1401 pour la deuxième génération et le 360/30 pour la troisième génération.

Chacune à son époque (et de nombreux 360/30 sont toujours en service) a représenté la machine de gestion type de classe *moyenne-basse*, c'est-à-dire non pas un bas de gamme, mais la première machine «sérieuse» du catalogue destinée à des entreprises qui ne sont plus de PME. Des systèmes 1401 au 360/30 en configuration normale étaient loués environ 50 000 F par mois (Francs courants), ou vendus un peu plus de deux millions de F.

Pour comparer la rapidité de deux unités centrales, on peut réaliser une savante pondération entre les temps d'exécution des différentes instructions en fonction de la nature des applications. On obtient ainsi les *indices Gibson mix* en gestion et en scientifique. Ou alors, si l'on n'est pas trop perfectionniste, on compare pour le scientifique les temps d'addition flottante en binaire, et pour la gestion les temps d'addition de chaînes décimales... et on ne tombe pas très loin des Gibson !

Ayant perdu notre formule de calcul du temps d'addition pour le 360/30, un coup de fil au saint



Le traitement de chaînes en 8080 : facile...

des saints nous renseigne : 81 microsecondes pour 16 ch + 16 ch, soit 2,66 fois plus rapide que le 8080 normal (qui existe aussi en versions de course). Ce n'est pas du tout ridicule.

Voyons pour le 1401 : le temps d'accès est de 11,5 microsecondes pour un caractère décimal de 6 bits. Pour réaliser l'addition, il faudra 3 accès par chiffre (1 pour lecture du premier, 1 pour lecture du second, 1 pour stockage du résultat) ce qui donne donc un temps minimum de :

$$11{,}5 \times 3 \times 16 = 552 \text{ microsecondes}$$

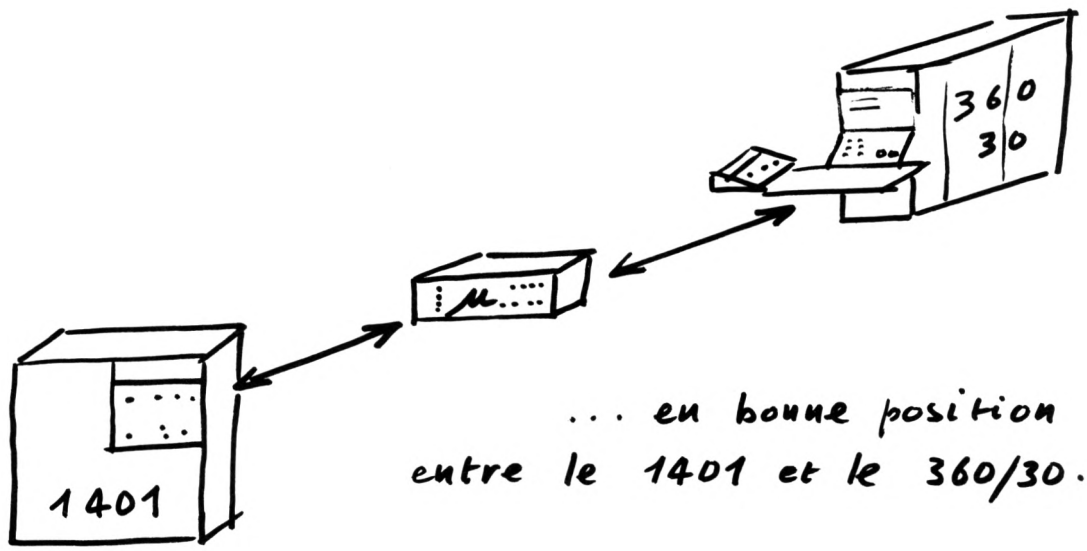
ce qui est 2,56 fois plus lent qu'Alcyane.

Battu le 1401 ! Et surtout, notre mémoire centrale (à octets, elle) peut être beaucoup plus importante que les 16 K caractères du 1401. Le 360/30, lui, est pendant de nombreuses années monté au maximum à la même taille de 64 K octets.

Le résultat est certes saisissant. Alcyane se place en bonne position entre le 1401 et le 360/30 pour la puissance de l'unité centrale... qui pèse moins de 10 kg ! Peu de comparaisons permettent de mettre en évidence de façon plus saisissante les progrès de l'électronique.

Le lecteur aura sans doute compris à nos comparaisons que nous avons gardé une certaine nostalgie — justifiée pensons-nous — pour les caractéristiques de certaines machines, et tout particulièrement du 1401, dont l'un des seuls défauts était une mémoire limitée. Il est encore temps d'y réfléchir, et nousreviendrons sur ce point un peu plus loin.

Muni de ces chiffres, nous allons maintenant faire un pas de plus : les reporter sur le graphique que nous avons élaboré pour le chapitre 1 de «*La micro-informatique : introduction aux systèmes*



... en bonne position entre le 1401 et le 360/30...

répartis» (22). Première question : quelle vitesse moyenne allons-nous attribuer à l'Alcyane pour son Gibson mix de gestion ? On ne peut pas la déduire facilement du temps obtenu, une addition 16 + 16 étant nettement plus longue que la moyenne en gestion (un peu plus de deux fois). Fort heureusement, nous sommes arrivés en bonne place entre le 1401 et le 360/30 : prendre la moyenne des deux semble raisonnable, ce qui nous donne 10 000 instructions/seconde.

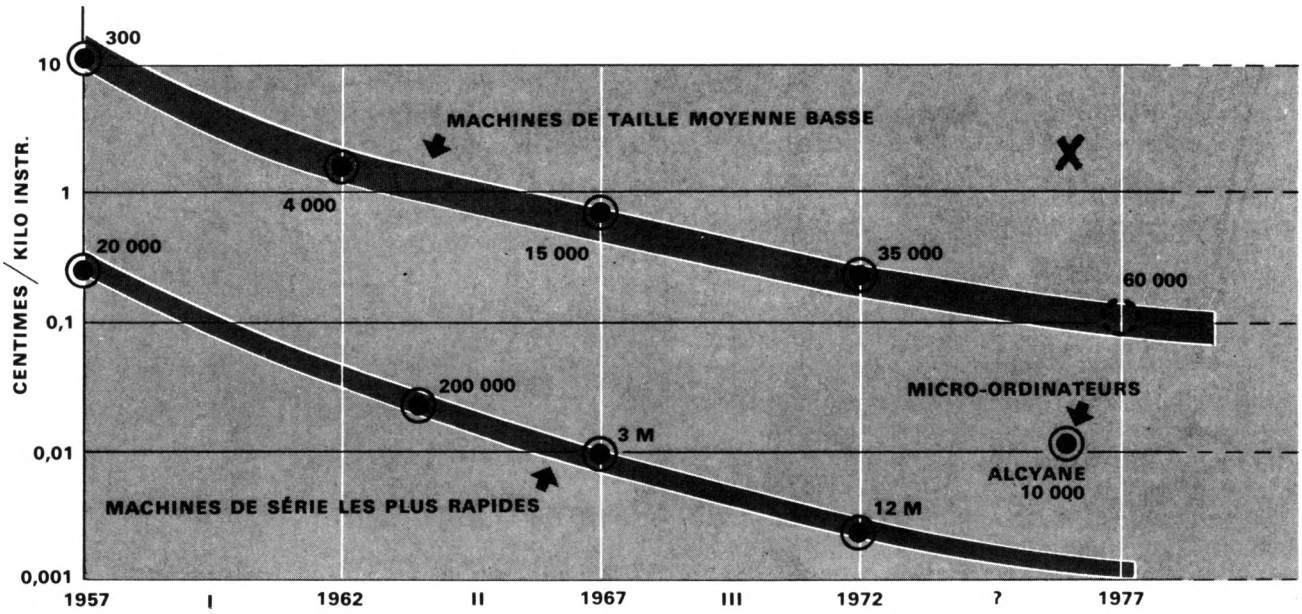
Voyons maintenant les coûts. Un système de gestion normal avec clavier, visualisation TV, deux disques souples et une imprimante à aiguilles à 100/150 caractères seconde revient à l'achat à environ 40 000 F. Ramenons à un coût mensuel d'amortissement de 1/40e, sensiblement comme pour les autres machines, soit 1 000 F (ou 10_5 cmes). Il vient : (10_5 cmes x 1 000 i) (182 h x 3 600 s x 10 000 i/s) = 0,015 l'unité étant le centime par millier d'instructions exécutées.

On voit où se situe le point sur le graphique : Alcyane, la première machine européenne que son coût, sa technologie et sa puissance font rentrer dans la case jusqu'ici vide des *micro-ordinateurs* est effectivement représentative d'une classe nouvelle de matériels, qui n'obéit visiblement à loi de Grosch : son point représentatif calculé d'après elle devrait se trouver au voisinage de la croix qui est à droite dans le rectangle supérieur...

Plus loin, dans le texte du chapitre 1, au moment où nous analysons d'un peu plus près la loi de Grosch et la loi de Bouhot en concurrence, est discuté le problème des entrées/sorties sur lequel nous aimerions revenir. Un graphique tel que celui que nous venons d'étudier favorise indûment les machines très puissantes, dont en fait le débit est limité par les entrées/sorties disponibles dans le cas de travaux de gestion, ainsi que par le temps de déroulement du software complexe qui les gère. La courbe des machines de série les plus rapides n'est pas représentative de leur débit réel, et remonterait de plus d'une division par rapport à celle des machines de taille moyenne basse (MTMB) si l'on faisait intervenir ces facteurs.

Si l'on prend à l'inverse uniquement les rapidités d'entrée/sortie pour calculer les débits, on obtient un deuxième critère de comparaison intéressant à opposer au précédent. Essayons-le. Une MTMB est usuellement munie d'une imprimante 132 positions à 1 000 lignes/minute, et Alcyane en a ici une à 60/100 lignes/minute (une imprimante

(22) En collaboration avec Bruno Lussato et Bruno France-Lanord, 280p., septembre 1974, Editions d'Informatique, 82 rue Lauriston, 75116 Paris. Nous ne pouvons reprendre ici tous les détails de calcul du graphique cité et des suivants.



Evolution au cours des années du coût de l'instruction exécutée pour deux catégories d'ordinateurs. En haut, machines de taille moyenne ; en bas, machines de série les plus puissantes.
Les vitesses pratiques sont indiquées en instructions (ou en millions d'instructions) par seconde. Les coûts sont en centimes par mille instructions. Les chiffres romains indiquent les trois générations de machines. L'avènement des micro-ordinateurs invalide visiblement la loi de l'économie d'échelle.

à tête mobile saute à la ligne suivante quand il reste des blancs, et imprime souvent dans les deux sens). Le rapport est de l'ordre de 30 kilo-octets/s, soit un rapport encore du même ordre, et il en est à peu près de même pour leur temps d'accès. Le micro-ordinateur est donc environ 15 fois plus lent en entrée/sortie qu'une MTMB contemporaine, et 5 fois plus lent en traitement interne. Ces estimations doivent maintenant être corrigées de divers facteurs. D'abord, le micro n'a de simultanéité entrée/traitement/sortie qu'au prix de complications qui nous paraissent l'éloigner d'une vocation souhaitable de simplicité, mais n'oublions pas que nous comparons une batterie de micros dispersés avec le travail des partitions individuelles dans la MTMB, et que sa simultanéité est à apprécier à ce niveau. Ensuite, il travaille préférentiellement en monoprogrammation (nous y reviendrons), et il n'a par conséquent pas besoin de système d'exploitation, d'où un rendement interne sans doute double. Autrement dit, il sera en pratique limité en débit par ses entrées/sorties, ce qui se voit nettement en comparant avec une MTMB. Tout dépend alors du mixage des traitements, et du temps moyen de travail au clavier chaque jour par rapport au traitement autonome sur fichiers.

La comparaison n'est pas simple. Comme nous le disait un jour un de nos collègues américains *«je ne connais pas de problème complexe qui, si on l'aborde sérieusement et rationnellement... ne devienne encore plus compliqué»*. En gros, le micro coûte 50 fois moins cher et a un débit qui selon les critères (et la nature des traitements) est de 3 à 30 fois plus faible que la MTMB. Le moins qu'on puisse dire est qu'il ne suit absolument pas la loi de Grosch, et il semble raisonnable d'affirmer qu'il la prend même à contrepied : c'est la loi de Bouhot, et vous remarquerez que prudemment nous n'avons voulu y introduire aucun facteur de proportionnalité. Par contre, elle comporte une condition portant sur le software, condition qui nous paraît avoir son importance : la monoprogrammation monotâche.

3 - 9 - 2. De la simplicité

On sait que les ordinateurs de troisième génération représentent un grand progrès par rapport à ceux de la seconde. Ou du moins beaucoup de gens le disent (ce n'était pas précisément l'avis d'Amdahl et de ses collaborateurs, au moment de la fameuse bataille interne IBM de l'équipe des 1 400 contre celle des 7 000 scientifiques : c'est cette dernière qui a gagné, et nous a amené le 360 ; relisez la relation de la bataille dans les deux fameux articles de *Fortune* de l'époque, c'est fascinant).

L'un des facteurs importants de ce progrès était l'architecture des machines en ce qui concerne

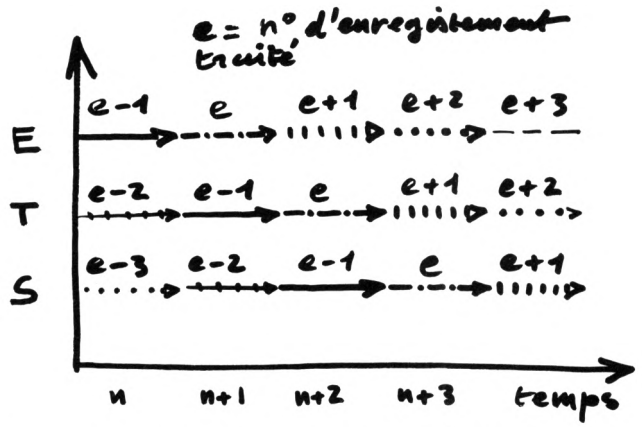


Diagramme très simplifié de la simultanéité entrée/traitement/sortie

les entrées/sorties, les dotant de la simultanéité ETS, entrées/traitement/sortie. Notre croquis en rappelle les principes : ce qu'il ne dit pas, ce sont les conséquences.

Vers les années 60 et quelques, je faisais partie d'un petit groupe studieux de prospects qui dans une pièce donnant sur la place Vendôme suivait un cours sur la nouvelle merveille IBM, le 7070 qui allait arriver... à peine 10 ans avant le 8080 :

$$8080 - 7070 = 1010_2 = 10_{10}$$

(vous voyez que c'était écrit dans les chiffres).

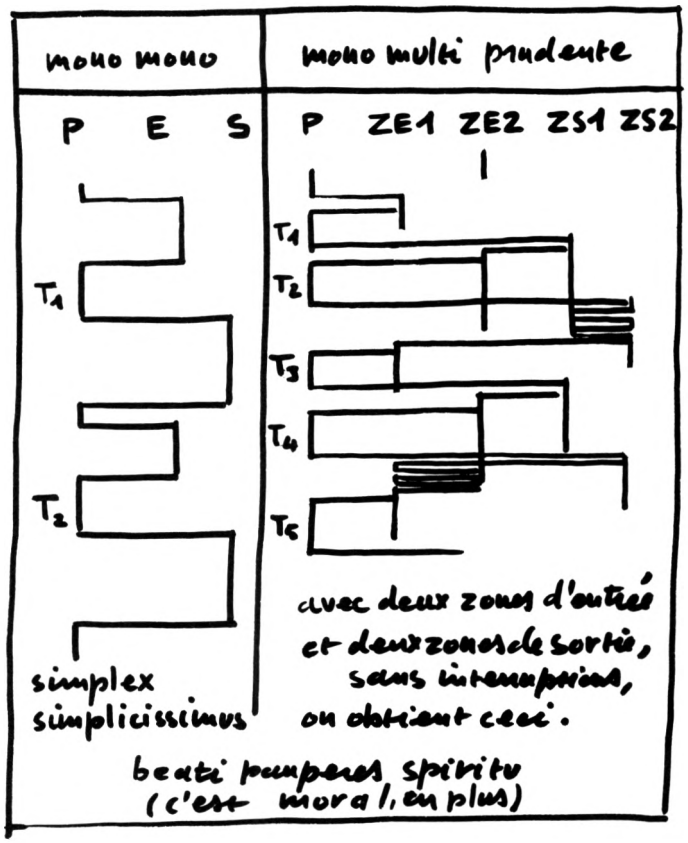
Bref, ce 7070 était formidable. Il avait trois accumulateurs ! Tout décimal, impeccable. A mots, certes, mais avec de bonnes astuces de découpage. Au bout de quelques jours, ayant assez fait le tour de la bête, comme nous avions tous pas mal d'années derrière nous de programmation laborieuse en langage machine et autres, nous avons attaqué un projet en commun : écrire un modeste programme. Je vous le donne en mille : lire une carte 80 colonnes et la cracher sur l'imprimante. C'est sérieux, je ne galèje pas.

A la fin de l'après-midi, les problèmes soulevés devenaient de plus en plus sérieux. L'instructeur était devenu complètement sec, et il nous promit pour le lendemain matin le responsable national de la maintenance du matériel. A la fin de la matinée nous étions en train de tourner tous en rond dans les canaux, entrecroisant nos priorités, pourchassés par l'horloge de démasquage des interruptions, mitraillés par ces dernières... et nous sommes unanimement arrivés à la conclusion que nous étions incapables d'écrire le programme. Bien sûr, en 1401 ce programme se serait composé de trois instructions écrites en 20 secondes.

La seule méthode en 7070 était (ce que nous avions voulu éviter) d'utiliser un gros programme qu'on logeait à demeure en mémoire centrale, l'IOCS (Input-Output Control System, système de commande des entrées/sorties). Cet IOCS était

une vraie boîte noire. Pour lire une carte, on faisait un GET CRD ou autre, pour imprimer une ligne un PUT PRNTL ou approchant, sans du tout savoir ce qui se passait à l'intérieur. Je n'avais pas lu Marx, mais j'ai ressenti l'aliénation. Il n'était plus possible de se servir de l'ordinateur tout seul : je venais de ressentir le premier choc du sofware (on sait où on en est aujourd'hui), le prix payé pour la simultanéité entrées/traitement/sorties.

Or, au même moment, comme nous l'avons dit, la programmation en 1401 était d'une simplicité biblique. Après les affres du 205 Burroughs, le fameux Datatron, qui apparaissait à l'époque en toile de fond des films américains de science-fiction (il avait un pupitre impressionnant, mais un système de pistes rapides sur le tambour qui ne simplifiait pas particulièrement le travail) le 1401 était un réel soulagement. Plus de problème d'optimisation d'une mémoire centrale rotative. Plus de mots à découper, mais une machine à caractères, donc à longueur variable. Et toujours pas de simultanéité ! On ne déroulait bien sûr qu'un programme dans la machine (monoprogrammation),



mais surtout ce programme ne faisait *qu'une chose à la fois* (monotâche). C'est peut être gênant pour un homme politique en vue, mais c'est bien pratique sur un ordinateur moyen. Quand on arrivait sur une instruction de lecture de carte, par exemple, on restait bloqué sur l'instruction jusqu'à ce que la carte soit lue, son image disponible en mémoire. Au contraire, sur une machine à simultanéité, l'ordre de lecture est envoyé au périphérique, ce qui ne demande que quelques microsecondes, puis on passe à l'instruction suivante sans attendre. La carte, elle, ne sera lue qu'au bout de 50 à 100 *millisecondes*. On voit que l'organisation des programmes devient bien différente, et beaucoup plus complexe.

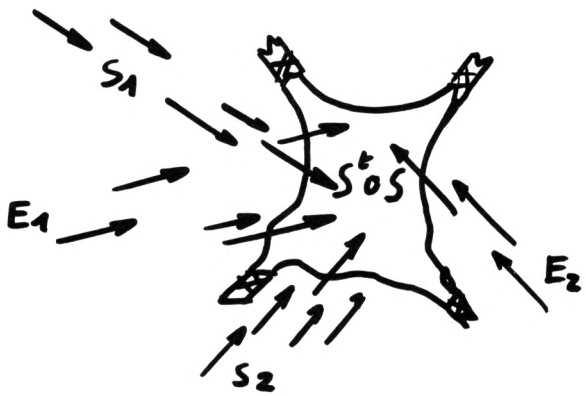
Certes, le 1401 possédait un peu de software : on recevait par exemple le s/p de lecture de bande magnétique, on l'examinait, on constatait qu'il contenait une foule de généralités parasites inutiles pour notre application, on le mettait au panier et on en écrivait un autre plus compact et plus efficace en un quart d'heure.

Pour corser les choses, une ingénieuse invention destinée à économiser du temps vit le jour : les interruptions. Disons-le tout de suite, nous n'avons rien contre les interruptions *internes*, ou *trappes*, manifestement très utiles comme nous l'avons vu, surtout sur une grosse machine complexe. Par contre, pour les interruptions *externes*, c'est autre chose, spécialement celles déclenchées systématiquement par les périphériques à destination de l'IOCS (de l'OS maintenant). L'idée de base est simple. Une opération d'entrée, ou de sortie, a été lancée par une instruction (*pas* une instruction du programme de traitement : on a vu que les E/S se font uniquement par le biais de l'IOCS, du superviseur ou tout ce qu'on voudra, bref du software, et l'on n'a plus le droit d'en écrire une dans un programme ordinaire), puis on fait autre chose, et quand l'opération d'E/S est enfin terminée, le périphérique le signale en envoyant une interruption qui permet au superviseur d'être informé sans perte de temps de ce qui se passe.

Ce tableau est idyllique, un peu trop peut-être. Dans la réalité, un mitraillage continu d'interruption tel que l'inflige ce système, se traduit par une série de conséquences néfastes. Par exemple :

— il complique d'abord considérablement l'écriture, la mise au point et l'évolution ultérieure du software de base, et le fait foisonner,
— corollairement, il diminue la fiabilité dudit software, dont dépend par ailleurs le bon fonctionnement de tout le reste,
— il provoque une considérable dégradation du rendement général. En effet, on est sans arrêt amené à sauvegarder contenu des registres et état du processus, puis à les restituer, d'où un dilemme sérieux : au fur et à mesure qu'on multiplie le nombre de registres généraux et de drapeaux d'état d'un processeur pour le rendre plus efficace, ceci n'est valable qu'en monoprogrammation, car au contraire les temps de

sauvegarde/restitution de contexte augmentent, et l'efficacité en multiprogrammation décroît ! Ceci est grave — il n'y a qu'à voir *l'overhead* (23) des systèmes d'exploiration. Une méthode pour s'en tirer est alors de concevoir une machine avec n jeux de registres généraux, commutables automatiquement chaque fois qu'une nouvelle interruption se présente avant que la précédente ait fini d'être traitée.



.. le martyre de l'os ...

Mais qu'est-ce qui arrive si le nombre de niveau x dépasse n ? Et comment avoir à l'avance la moindre idée du nombre qu'on atteindra, compte tenu de la variété des situations réelles possibles ?

Lorsqu'on parle à un programmeur système de moins de 40 ans de machines sans interruptions de programme, et si par-dessus le marché c'est dans un environnement temps réel, il vous regarde avec des yeux ronds, ou les détourne pudiquement pour ne pas gêner un simple d'esprit...

Or, réfléchissons un peu à ce que fait réellement le superviseur à l'arrivée d'une interruption : de toutes façons, il doit bien d'abord aller s'informer pour savoir *d'où* provient ladite interruption. Ensuite, il n'est sûrement pas en mesure de lui faire correspondre une action immédiate : il faut par exemple laisser se terminer la séquence du programme de traitement en cours jusqu'à ce qu'elle ait lancé une opération d'E/S. Par conséquent, le superviseur va stocker quelque part l'indication qu'une interruption provenant de tel ou tel périphérique est arrivée, puis il va rendre la main au programme de traitement en cours.

En somme, les interruptions sont tout simplement considérées comme des messages émis par des processus externes variés à destination de processus internes qui les attendent... Alors, à quoi bon tout ce remue-ménage, et pourquoi ne pas tout simplement tester régulièrement des drapeaux d'état correspondant aux processus exter-

nes... ce qu'on faisait avant les interruptions ! Et pour faire plaisir à tout le monde, pour éviter de farcir régulièrement (ce qui serait d'ailleurs assez simple) nos bons programmes de traitement de CALL TOURDHORIZON à intervalles de temps à peu près constants, admettons une interruption de la classe 1 non nuisible : une trappe chronométrique que la carte adéquate d'Alcyane enverra pour vous inciter à aller voir s'il y a des lettres dans les boîtes.

Tout ceci n'est pas de la théorie, c'est du douloureusement vécu, et ce n'est sûrement pas ceux qui ont passé nuit sur nuit blanche pour des tests de programmes bourrés d'interruptions (puis hélas souvent après lors de l'exploitation...) qui nous jetteront la première pierre.

Il y a quelques années (une douzaine), l'un des rares ordinateurs de commutation de messages qui marchait comme un tracteur, hard et soft, et fut la base de plusieurs réseaux transcontinentaux de transmissions de données, était le Datanet 30 de General Electric. On pouvait connecter jusqu'à 128 lignes de transmissions directement sur l'unité centrale : voilà un des environnements les plus exigeants en matière d'interruptions. Or le Datanet 30 avait été conçu *sans aucune interruption*.

Il semble que la leçon se soit quelque peu perdue. Pas tout à fait cependant. Nous conseillons à ceux qui en lisant ces lignes se rendent compte qu'ils pensaient depuis longtemps comme nous, sans avoir eu peut-être l'occasion de le dire tout haut, d'étudier pour étayer leurs idées une des plus astucieuses machines conçues depuis pas mal de temps : le System 250, de Plessey. Voilà une machine conçue pour la commutation téléphonique et qui est dépourvue de toute interruption : or c'est un ordinateur multiprocesseur, avec un système d'exploitation gérant multiprogrammation et mémoire virtuelle. Qui plus est, ce qui

(23) Temps non productif du processeur, consommé par le système d'exploitation. A noter que overhead en langage courant signifie *frais généraux*...

est au moins aussi remarquable, il n'y a *ni état superviseur, ni instructions privilégiées.* Programmeurs système, voilez-vous la face, il n'y a plus qu'à tirer le rideau. Pour de plus amples détails, voyez les *Proceedings of the 1972 IEEE International Switching Conference,* tenue à Cambridge, Massachusetts.

3 - 9 - 3. Doublement à la portée de tous

Les micro-ordinateurs (le hardware) ne coûtent plus bien cher, et en ce sens seront bientôt à la portée de tous.

Mais qu'en sera-t-il de l'aspect logiciel ?

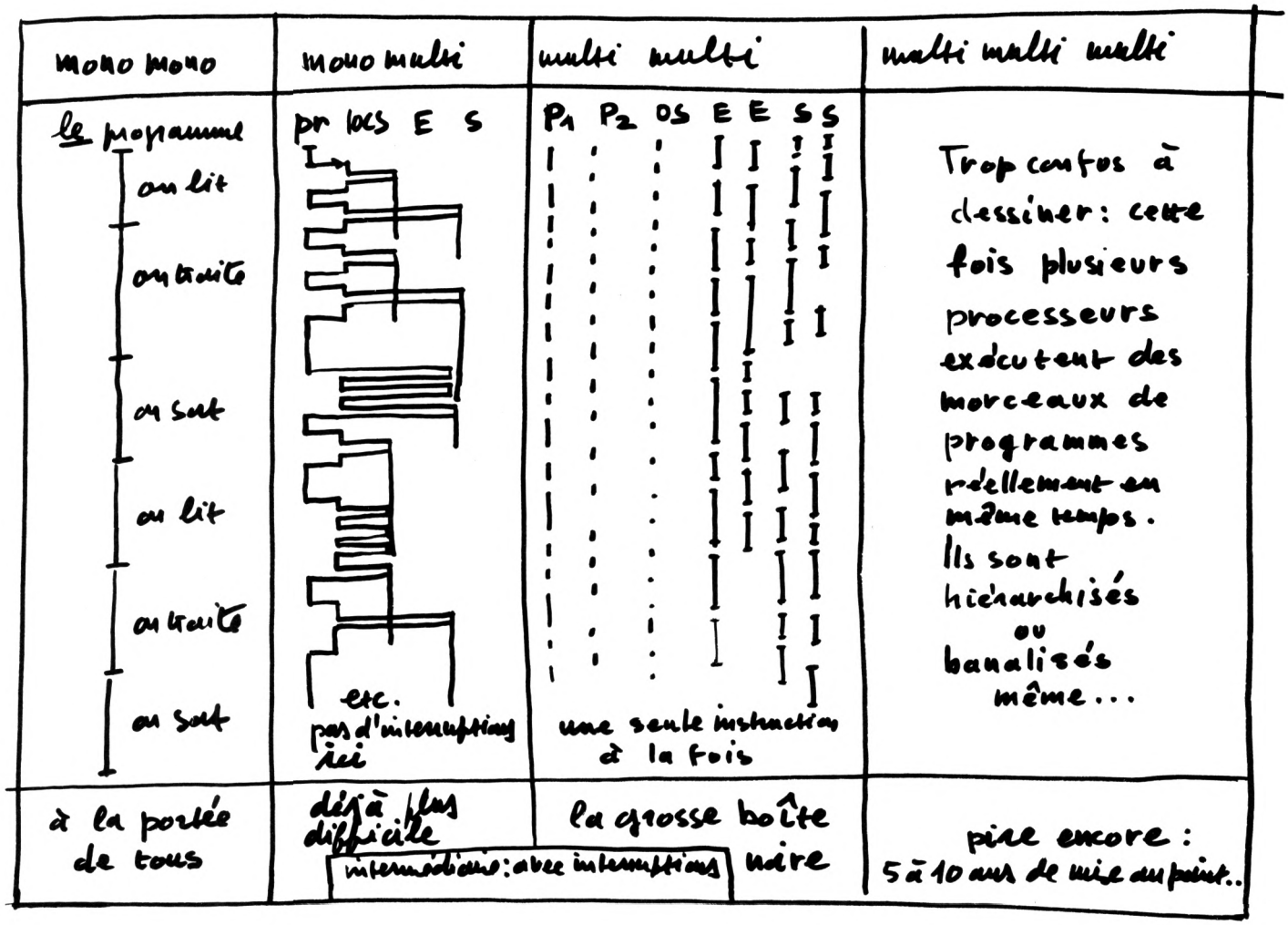
Nous tenons une grande chance avec les micro-ordinateurs, car nous allons pouvoir exercer notre métier de façon encore plus intéressante, et avec des clients enfin souriants : ne la démolissons pas, cette chance, et essayons de raisonner la tête froide, ce qui va nous fair découvrir que tout converge dans le bon sens.

Le hard n'est pas cher, donc. Par conséquent, il faut essayer de faire que le soft soit bon marché lui-aussi. Et comment pouvons-nous y arriver ?

Par la simplicité !

Nous allons donc travailler en monomono (et découvrirons tout à l'heure qu'on peut aller très loin de cette façon), faire par là même disparaître tout le soft parasite, et économiser beaucoup d'heures et d'argent. Quel dommage, direz-vous, il marche si vite ce 8080, on pourrait facilement y mettre cinq ou six claviers, c'est du gaspillage, on n'optimise pas.

Eh bien non, on n'optimisera pas. Et c'est justement parce que le hard n'est pas cher qu'on n'a plus besoin d'optimiser, pas plus que vous ne vous relayez avec votre femme pour faire tourner votre voiture autour du pâté de maisons au moins 12 heures par jour sous prétexte de l'optimiser : la comparaison commence à tenir, les prix se rapprochent de très près. Et c'est justement parce qu'on n'a plus besoin d'optimiser que le soft devient simple — et pas cher.



Finalement, par cette démarche, nous mettons l'ordinateur *doublement à la portée de tous :*
— par l'abaissement du coût,
— par l'abaissement de la complexité.

3 - 9 - 4. Simplement complexe

Est-ce à dire que nous allons être limités dans la complexité des problèmes qu'on pourra traiter ?

Non, car la complexité que nous avons réduite, c'est celle du soft de base : IOCS, superviseur, système d'exploitation, tout ce que vous voulez, bref, ce qui sépare l'honnête programmeur de sa machine. Le problème, lui, peut être touffu. Prenons un exemple : nous pourrons réaliser au clavier le traitement complet d'une police d'assurance, depuis le début de l'entrée des rubriques avec toutes les opérations de validation, intrinsèques et d'après fichiers, en passant par les calculs de primes, jusqu'à la sortie de tous les sous-produits et mouvement qui s'en déduisent. La taille globale du programme pourra faire 150 K, mais il consistera en une succession de segments recouvrants appelés de la diskette en *monomono,* c'est-à-dire en sachant toujours à chaque instant *où on en est et ce qu'il va falloir faire.*

Mettons bien les points sur le i : une telle organisation est sous le contrôle complet du programme et ne présente aucune difficulté pour le programmeur. Elle est absolument identique à l'appel de sous-programmes, à la seule différence près que les s/p sont en réalité des segments (des pages si vous voulez) de programmes stockés sur support magnétique externe et qu'on charge au moment de l'appel dans une zone mémoire banalisée réservée à cet effet. Si nous donnons ces précisions, c'est qu'ayant récemment mentionné la puissance et la simplicité du monomono vu sous cet angle, nous avons reçu en retour la question «est-ce que l'overlay (24) est géré ?» A la lumière des définitions précédentes, on voit que cette question — valable dans un environnement multiprogrammation avec système d'exploitation — n'a pas de sens ici : l'appel qui dans un programme normal s'écrit :

```
CALL        SSPROGN
```

est simplement remplacé en overlay monomono par

```
LXI         B,N
CALL        SEGMENT
```

(24) Overlay : angl. pour **recouvrement,** action de charger en mémoire un segment de programme qui « recouvre », en l'effaçant, un autre segment chargé antérieurement. La **zone d'overlay** est l'emplacement réservé en mémoire pour le chargement des segments.

Le s/p SEGMENT trouve dans BC la référence N du segment à appeler, qui sera tout bêtement son adresse physique sur diskette et sa longueur, il n'y a même pas de table à consulter. Le segment est alors chargé en mémoire, et on se branche à son début. Il finit obligatoirement par RET qui ramène juste après CALL SEGMENT, et la manœuvre recommence identiquement chaque fois que nécessaire.

3 - 9 - 5. de la macro ➡ à la micro

voir schéma p. 122

3 - 9 - 6. Un dialogue

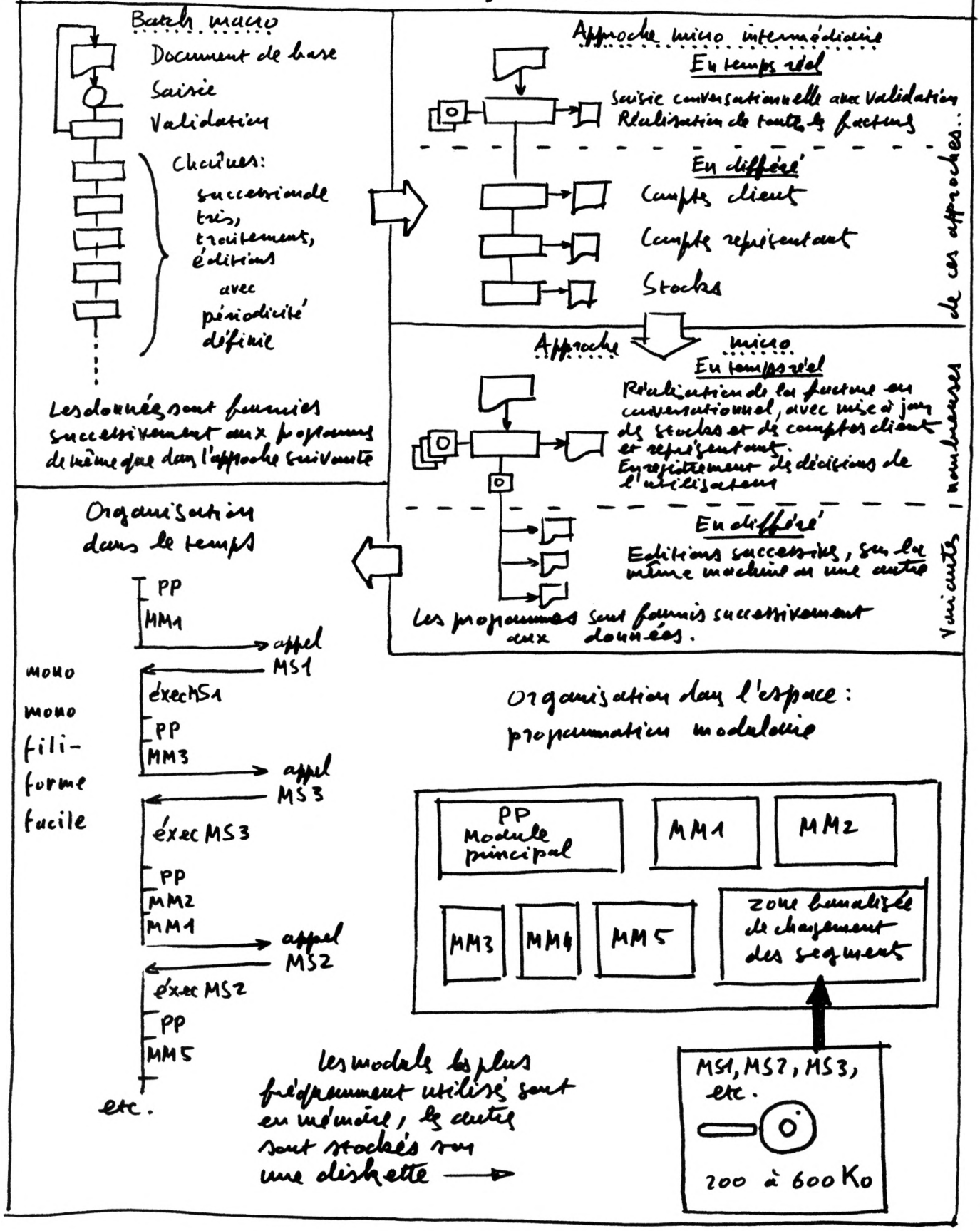
L'un des aspects les plus sympathiques des micro-ordinateurs est que l'on retrouve enfin pour l'écriture et la mise au point des programmes *la machine entièrement à soi.* Ce faisant, on vient à cette ancienne et si agréable méthode où l'on s'installe face à sa machine, pour examen des résultats intermédiaires, inspection de zone de mémoire, de contenus de registres, etc. On est ramené 20 ans en arrière au plaisir du travail au pupitre du 650, où l'on corrigeait immédiatement en mémoire les erreurs les unes après les autres, avant de «perforer le tambour», c'est-à-dire de sauvegarder sur cartes le programme sur lequel on venait de travailler. Donald Knuth, n'a-t-il pas dédicacé sa fameuse série *The Art of Computer Programming* (la bible) : «au 650 de l'Université de Stanford avec lequel j'ai passé tellement de soirées agréables»... Tous ceux qui ont pratiqué cette méthode l'ont amèrement regrettée le jour où les nécessités économiques de l'exploitation, dues à l'évolution des machines, ont fini par imposer partout la salle d'exploitation interdite d'accès aux programmeurs, le passage des travaux au guichet et les attentes interminables entre deux passages, forçant souvent à travailler sur plusieurs programmes en même temps pour ne pas perdre trop de temps.

Ici, non seulement ce mode est retrouvé, mais avec des facilités bien accrues :

— superbe visualisation sur écran au lieu de déchiffrer du code biquinaire sur les voyants,
— exécutions plusieurs dizaines de fois plus rapide,
— pas besoin d'optimiser des emplacements sur le tambour pour gagner du temps,
— stockage ou appel des programmes en secondes (diskettes) au lieu de minutes,
— coût ridiculement bas : personne ne viendra vous reprocher de gaspiller du temps machine si vous cafouillez quelque peu !

Les jeunes programmeurs vont réapprendre d'intéressantes techniques oubliées de mise au point, et d'autres nouvelles :

3-9-5. DE LA MACRO ⇨ A LA MICRO



— insertion de points d'arrêts avec vidages et manœuvres variées,
— s/p d'exploration de mémoire cherchant une configuration particulière de bits (adresse suspecte, message alphabétique, etc.),
— visualisation en double format ; hexa et symbolique retraduit sur place, se complétant,
— comparaison de zones mémoire avant et après exécution pour détection des différences,
— utilisation d'assembleurs interactifs avec symbolique stocké en mémoire.
etc, etc.

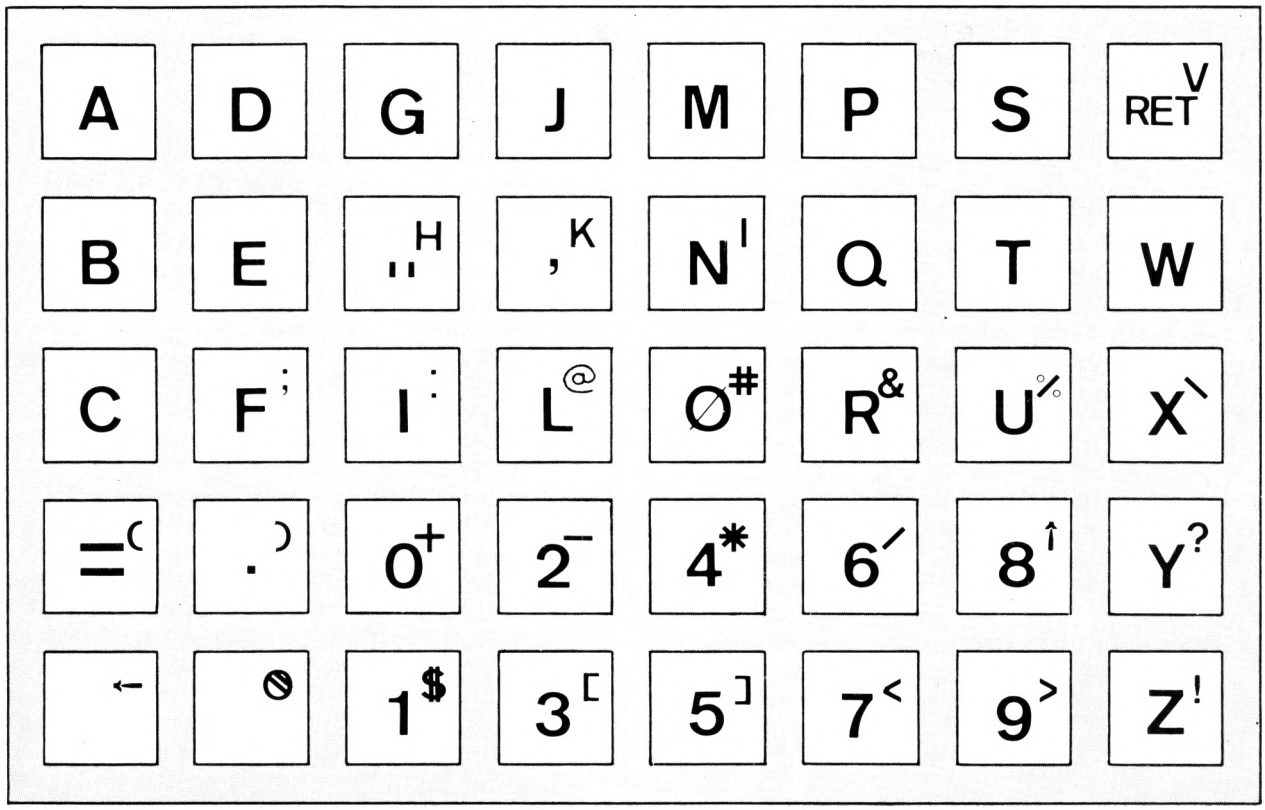
Enfin, et c'est l'essentiel, nous disposons de langages de programmation évolués *interactifs*. Programmer en BASIC ou en APL, c'est *engager un dialogue avec la machine.* Développés pour le travail à la console de temps partagé (télétype en général), ces langages conviennent à merveille au micro, puisque par principe il est entièrement à votre disposition, en temps réel, pour vous aider.

Dès la ligne de BASIC tapée, la machine répond : elle exécute, ou signale vos erreurs de syntaxe, selon le mode. La programmation en est entièrement transformée. Au lieu de s'échiner sur du papier à lire ou à relire, puis finalement à attendre des heures un passage machine qui revient bourré de bêtises dues à des erreurs de perfo des cartes programme, on va directement à son clavier entrer les lignes, et elles s'affichent sur la visu avec les diagnostics. On peut à tout moment corriger, effacer, insérer lignes ou caractères, exécuter des morceaux, mettre à jour, lister, etc. Enfin, on peut insérer des routines en assembleur qui permettent à bon compte d'augmenter la rapidité de traitement.

3 - 9 - 7. Le B, A, BA des claviers

Et comme bien peu dépassent, comme le montrent les statistiques, le niveau de la frappe avec deux doigts sur un clavier (c'est mon cas, et probablement le vôtre), l'un des claviers d'Alcyane a été dessiné pour les *dactylographes occasionnels :* il est *en ordre alphabétique,* avec 10 touches numériques en frappe directe, et un arrangement des 64 symboles USASCII choisi au mieux pour le travail en BASIC (l'autre clavier est un AZERTY).

Pourquoi du reste utiliser les claviers QWERTY ou AZERTY classique ? A cette question vient la réponse «parce qu'ils sont optimisés pour la rapidité» (ce qui du reste n'aurait guère d'importance pour un travail de programmation). La réalité est malheureusement bien autre ! Lorsque l'ingénieur américain qui conçut la première machine à écrire pratique vers 1870 étudia la répartition des caractères sur le clavier, il le fit effectivement de manière à obtenir une bonne rapidité de frappe. Mais dès que les premières machines furent livrées en clientèle, catastrophe ! Les utilisatrices



tapaient plus vite que lui, et les marteaux s'accrochaient tous entre eux...

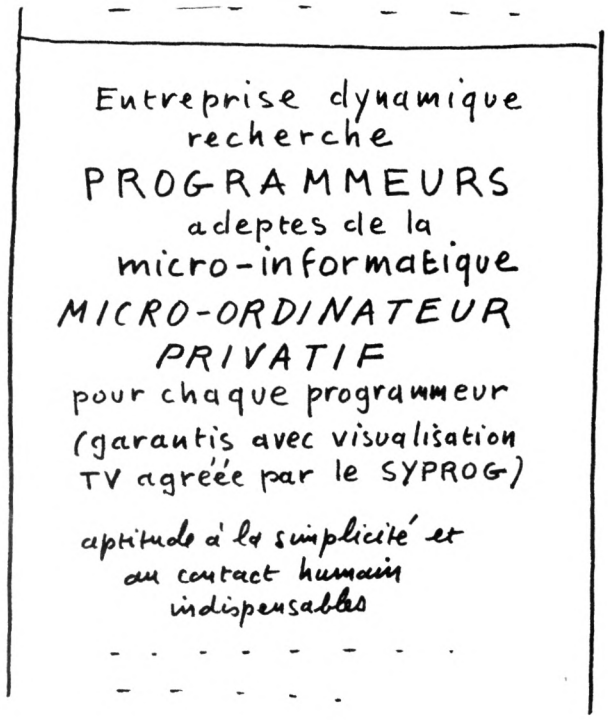
Ne pouvant accélérer la rapidité du mécanisme, il ne lui restait plus qu'une solution désespérée, et c'est celle qu'il adopta : vous la sentez venir ? Eh bien oui, il redistribua les caractères de *façon à freiner la frappe autant que possible !*

Et c'est cette disposition qui est devenue le standard international...

Cette histoire n'est pas une galéjade, elle est parfaitement sérieuse (rappelez-vous au passage que le dessinateur chargé du dessin des cames d'ouverture et de fermeture des contacts pour l'envoi sur ligne telex des caractères Baudot, en mode asynchrone, s'était trompé dans son calcul d'angle de came, et que c'est à la suite de cette erreur que le bit stop a la bizarre valeur de 1,52 fois le temps d'un bit normal : là encore cette valeur est devenue norme internationale !).

Il existe un clavier vraiment optimisé pour la vitesse : c'est le clavier Dvorak, et il fait l'objet d'un classement à part dans les concours internationaux de rapidité de frappe à la machine à écrire, tant sa supériorité est grande. Je me suis toujours demandé pourquoi il n'avait pas été adopté pour les matériels de saisie, où les opératrices/teurs sont formés en général avant d'avor appris les claviers machines classiques.

Pour en revenir à notre micro, le travail en interaction directe, avec visu, sur un programme unique au lieu d'une alternance, est vraiment le rêve pour tout programmeur. Responsables informatiques, tenez-en compte. Les programmeurs y gagnent en agrément du travail, et vous en productivité de l'équipe en programmes testés. Sans doute verrons-nous dans quelques années, dans le style ampoulé auquel nous ont habitués les promoteurs immobiliers, des petites annonces du même parfum que celle reproduite ici...



3 - 9 - 8. Informaticiens et micro-informatique

Contrairement à beaucoup de personnes, nous pensons que l'ordinateur est l'ami de l'homme. Enfin peut-être pas sous toutes ses formes, mais assurément en ce qui concerne notre vieux 650, et encore plus le micro.

Cette affirmation demande sans doute à être étayée. L'homme, ici, c'est d'une part l'utilisateur (laissons lui la première place), et d'autre part l'informaticien (nous, autrement dit).

En ce qui concerne l'utilisateur, je couperai au plus court en reproduisant un récent éditorial qui résume brièvement mon opinion sur le sujet :

DECENTRALISATION/CENTRALISATION, OU MIEUX ENCORE ?

De tables rondes en journées, de séminaires en congrès, les systèmes répartis et la micro-informatique, après une réaction initiale de rejet, sont finalement devenus une tarte à la crème. Et certes, ce n'est pas nous qui bouderons notre part.

Cependant, une bonne partie de ces discussions fait fausse route : elles tournent autour de centralisation/décentralisation, ou plus modestement (et plus immédiatement) de concentration/déconcentration. Cela est bel et beau, mais suppose de grandes manœuvres (ne parlons pas de conflits) du ressort des directions générales, étalées sur 4 à 8 ans, et laisse de côté l'essentiel : l'informatique des opérationnels, absente à l'appel.

Car enfin c'est là la grande faillite de l'informatique. A 90 % elle est faite par des gestionnaires pour des gestionnaires, avec un beau mépris ou une belle ignorance de ceux qui sont derrière un comptoir, aux prises avec les clients, à la facturation, ou à l'entrée du magasin de l'usine devant des casiers. A la décharge des informaticiens, les matériels se prêtaient bien mal à l'aide directe.

Qu'est-ce donc, au fait, que notre micro-informatique ?

Des traitements locaux spécialisés multiples en conversationnel, avec consolidation centrale, apportant à l'homme une aide immédiate et agréable dans son travail quotidien, et donnant à l'information un maximum de valeur ajoutée dès sa création.

Là, l'aide aux opérationnels est claire. Il fallait voir, lors d'une séance de définition de logiciels

destiné à des concessionnaires automobiles la joie des responsables de vente de pièces détachées devant un outil comme fait pour eux !

Ce sont les SSCI — nous devons le dire — qui vont matérialiser cette nouvelle informatique dans de nombreux secteurs.

Souhaitons que les informaticiens d'entreprise secouent aussi le joug : légion sont les opérationnels qui les remercieront pour ces myriades de petites machines sociables, enrichissant le travail et facilitant la vie de tous les jours.

J.-P. B.

En ce qui concerne l'informaticien (et je me classe certes sans aucune honte dans cette catégorie), la micro-informatique apporte un avenir absolument fantastique d'opportunités et d'intérêt du travail.

J'avoue avoir été assez désolé de la façon dont la chose a démarré, avec des polémiques journalistiques assez ridicules qui font qu'aujourd'hui encore la micro a plutôt mauvaise presse chez certains d'entre nous. Il est temps d'essayer d'y voir clair : la réalité est tout autre que l'avenir sombre qu'ils imaginent.

Tout d'abord, la micro va permettre de rendre un meilleur service aux utilisateurs comme dit plus haut : c'est là un fait essentiel, qui nous donne l'opportunité de cesser d'être les mal-aimés dans de nombreux secteurs. Mettre en place chez les gens, les aider à utiliser des machines qui leur rendent très directement service, n'est-ce pas un travail éminemment satisfaisant ?

Au lieu d'être pris dans des carcans d'OS et de Bases de Données, de Télécomm, dans lesquels on se débat péniblement en étouffant à chaque instant, écrasés par les uns et les autres, n'est-il pas libérant et oxygénant de configurer des micros en les adaptant, matériel et programmes, au mieux de la satisfaction des utilisateurs, toujours en recherche de simplicité et de service ?

Il va falloir faire de l'organisation, qui est *avant tout un élémentaire travail de bon sens sur le terrain*. A ceux qui en ont peur, disons qu'il existe des cours (celui du SCOM, et bien d'autres), des ouvrages, et que toutça *est très simple* pour un programmeur navigant dans ses ordinogrammes jour après jour. Et, surtout, ne pas oublier que c'est l'utilisateur qui est souvent la meilleure source d'idées pour un travail qu'il connaît à fond : il suffit de l'en faire accoucher en discutant avec lui.

Un mot de mise en garde : il ne faut pas travailler dans l'état d'esprit «grosses chaînes automatisées». Prenons un exemple simple, celui de la tenue de stock en liaisons avec la prise de commandes. Dans le macro-ordinateur, tout doit défiler en continu sans arrêt, tout doit donc être programmé. Ici, en micro, la chose se passe tout à fait différemment. Supposons qu'arrive une commande, et que sur un de ses postes le stock soit trop faible pour la satisfaire : signalons simplement le fait sur l'écran, et *laissons l'homme prendre une décision.* C'est un travail qu'il a fait des années de sa vie, gardons-lui le choix de l'une des très nombreuses combinaisons possibles :

— différer la livraison du poste
— en livrer une partie, différer le reste
— livrer tout ce qu'il y a en stock
— passer une commande de réapprovisionnement
  • normale pour un certain nombre,
  • en urgent peut-être pour tout, ou pour une partie,
  • à un autre dépôt,
  • etc, etc.

Nous nous bornerons alors à *enregistrer* les éléments de sa décision. Les programmes résultants seront beaucoup moins complexes que ceux du macro, et l'analyse aussi ! (que dire du cas d'une paye centralisée devant prendre en compte toutes les particularités locales...).

Autre exemple de pensée micro, toujours sur le même problème. Pourquoi, au lieu du format bien classique de l'enregistrement stock en macro, n'adopterions-nous pas quelque chose de plus sympathique ? ... mais oui ! la bonne vieille *fiche de stock* avec son aspect historique plus intéressant (25), qui plaîra beaucoup à l'utilisateur.

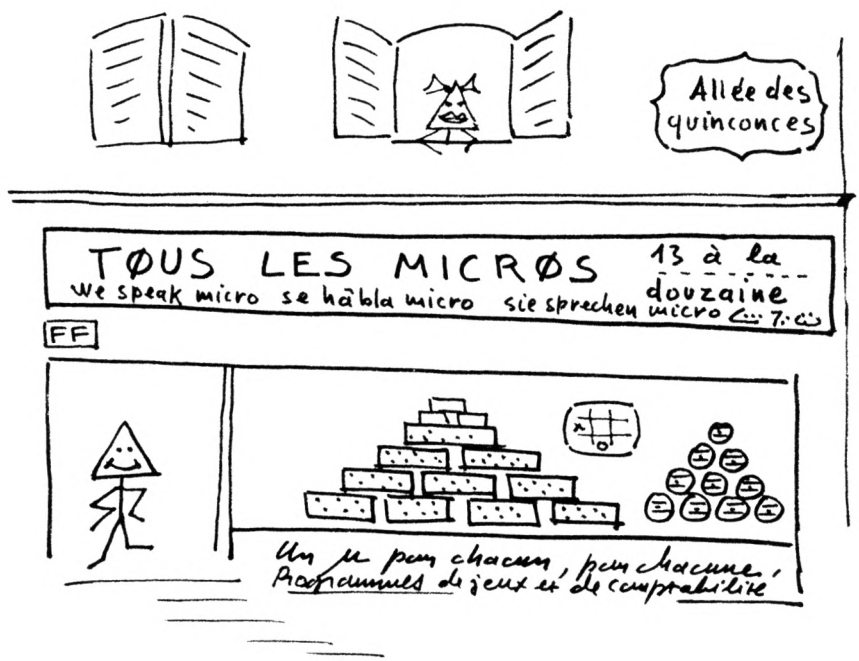
De même, on n'est pas forcé de multiplier considérablement le nombre d'unités de diskettes sous prétexte d'avoir *tout* en ligne. Dans presque toutes les applications, on peut découper les fichiers sur des critères tels que : type, emplacement géographique, vacation de traitement (matin/après-midi), marque, période, etc. Et n'oublions pas qu'au contraire d'un dispac, une diskette se change en quelques secondes. Du reste, la capacité d'une diskette passe de 300 000 à 600 000 octets, et 1 200 000 un jour, ce qui n'est pas si mal.

C'est dans cet état d'esprit qu'il faut raisonner : *simplicité et convivialité.*

Mais voyons un cran plus loin. Ces micros sont caractérisés par :
— un faible coût unitaire
— une puissance notable
— une structure simple en modules permettant un dépannage premier échelon très facile.

Il s'en suit que des opportunités phénoménales s'ouvrent à l'échelon individuel ou de très petits groupes d'informaticiens. Soyons réalistes, à un tel prix de machines, il va y en avoir partout, et elles ne vont y aller, ni s'y installer, toutes seules. Il y a donc un marché extrêmement diversifié et fructueux qui s'ouvre. Enumérons quelques idées :

(25) Voir Un Fil d'Ariane, Des Fins et des Moyens, *L'Informatique nouvelle*, n° février 1972 à avril 1973.



Un top-downa qui a fait cavalier seul...

— un informaticien faisant occasionnellement de la formation à l'informatique, acquiert un micro sur lequel il fait travailler des gens. Le succès est tel que le supplément de revenu lui rembourse le micro en 4 mois,

— un informaticien aide son beau-frère à développer un système simple avec un micro dans un cabinet d'assurance : le marché pour eux se présente comme énorme, il faut surtout qu'ils soient raisonnables pour la couverture géographique,

— un informaticien en a assez du dodo/métro/boulot de la région parisienne. Il installe dans une ville de province pleine de soleil de petits systèmes chez des commerçants détaillants ; il fait tout : montage des kits, programmation, installation, dépannage par substitution (il a chez lui un «stock» de trois modules de chaque type, deux floppys et une imprimante) en attendant le dépannage fin fait à Paris par le fabricant du micro (envoi postal) et sur place par les réseaux des fabricants de floppys et d'imprimantes,

— un informaticien a convaincu ses patrons de monter dans la société une activité supplémentaire : ils ont installé trois micros pour leur propre usage, et maintenant ils vendent des systèmes analogues clés en main avec modifications à la demande des entreprises du même secteur.

... La liste pourrait s'allonger à l'infini. Ce n'est pas étonnant. La seule liste des applications d'ordinateurs que nous connaissions (celle publiée annuellement par *Computers and People)* compte actuellement 3 000 titres, dont les trois-quarts au moins sont à la portée des micros.

Mais ces derniers, vu leur rapport performances/coût très accru, leurs dimensions physiques et consommation électrique très réduites, l'absence de climatisation, peuvent également aborder un grand nombre d'autres applications non listées là !

A ce point, il ne reste plus qu'à passer aux actes : les micros, il faut arrêter d'en parler en accumulant les promesses, et passer à leur concrétisation.

CHAPITRE 4
DU MICROPROCESSEUR AU MICRO-ORDINATEUR

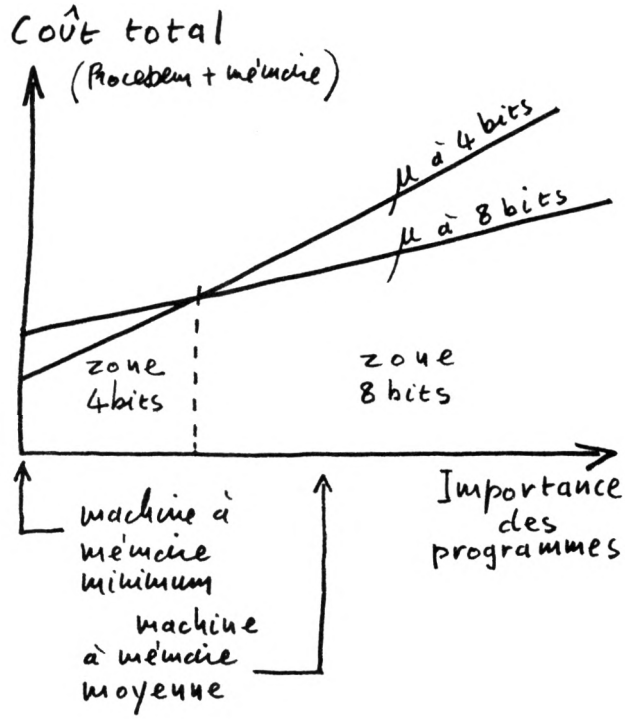
Il est long le chemin qui mène du microprocesseur au micro-ordinateur. Long, mais passionnant, et c'est cela qui nous a soutenus pendant le parcours que nous vous avions promis. Cet intitulé de chapitre, nous l'avons emprunté à l'ami Henri Lilen : il l'a pris comme titre de son intéressant ouvrage publié aux Editions Radio, et nous disait récemment que 80 % des projets à base de microprocesseurs démarrant dans les sociétés n'aboutissaient pas. Ce n'est guère étonnant, car il faut vraiment s'accrocher. Au bas mot, la conception et la réalisation pratique de l'unité centrale d'Alcyane nous ont pris des dizaines de mois-ingénieurs et transformés en homme-orchestres, ou plutôt en outres gonflées à éclater de problèmes de choix de conception, d'architecture, de composants, de périphériques, de logiciel, de travail minutieux à l'oscillo rapide, de problèmes de ruptures de stocks (pourtant gérés par macros !), d'usinage, de coûts, de pliages, de dessin, de sérigraphie, de visserie, d'emballage, ... et de documentation !

4 - 1. PER ARDUA AD ASTRA

Un an et demi après le coup de canon de départ, avec parfois du louvoyage sur place contre le courant, mais quelques bons bords aussi, nous avons passé la ligne d'arrivée : en mars 1976, la première série d'unités centrales était livrée, avec 10 semaines de retard, et tous les utilisateurs terminaient sans encombre le montage de leur kit (livré, il faut bien le dire en quatre ou cinq étapes...). Tous les périphériques du premier niveau tournaient au banc peu après, mais beaucoup de travail restait encore sur la planche ! Nous y sommes toujours, et de bon cœur.

4 - 1 - 1. Conception

Au chapitre 3 - 6 ont été évoquées les difficultés du choix du microprocesseur : tout le problème de la décision en avenir incertain, traité en séminaire avec JT quelques années avant ! Les types, prix et puissances des microprocesseurs et des mémoires étaient si évolutifs qu'il était peu aisé d'y voir clair. Le graphique reproduit nous permet d'éliminer les 4 bits. Les 16 bits n'avaient aucun avantage décisif, et surtout nous découvrions que, comme nous l'avons longuement détaillé, les «8 bits» étaient en réalité à la fois des 8, des 16 et des 24 bits, ce qui régla la question, vu qu'en plus ils étaient bien moins coûteux.



Vint ensuite l'objectif général d'utilisation du matériel : il nous déplut à tous d'envisager une machine de hobby, construite au prix le plus bas en sacrifiant la qualité des constituants, et par là-même d'une fiabilité approximative. Aussi, d'un commun accord, nous décidâmes fin 1974 :

— de réaliser un design totalement modulaire autour d'une structure de bus, par opposition avec une unité centrale figée sur une grande carte, globalement moins chère, mais difficile à étendre en possibilités,

— d'utiliser partout les composants de la meilleure qualité, tout en essayant d'abaisser les coûts par l'utilisation systématique d'éléments standards du commerce avec seulement les passes d'usinage indispensables (coffret),

— d'adopter systématiquement, sauf pour la mémoire comme on le verra, les composants les plus intégrés possibles, même au dépend du coût (cinq ou dix boîtiers TTL sont souvent moins chers que le DIP intégré unique qui les remplace) de façon à :

- simplifier au maximum le montage du kit par diminution du nombre de composants,
- diminuer l'encombrement,
- diminuer la consommation,
- et surtout atteindre un maximum de fiabilité.

— de diminuer le plus possible, pour les mêmes raisons, le câblage à réaliser (par exemple, toute l'alimentation est sur circuit imprimé) et là où il était inévitable, de concevoir les circuits imprimés de façon à remplacer le câblage classique en fil à fil, style plat de spaghettis, par des rubans de câble plat, bien plus nets et éliminant tout risque d'erreur (c'est du câblage structuré !),

— de bufferiser intégralement tous les modules, afin de protéger les MOS, permettre de supporter les extensions, et préserver des parasites,

— et surtout de concevoir un **système complet**. Derrière le mot kit peuvent en effet se cacher beaucoup de choses :

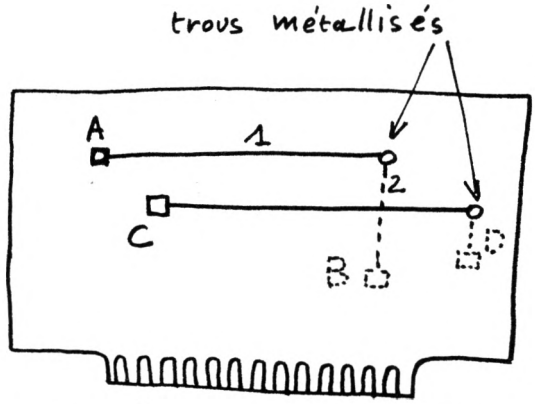
- le kit «fabricant de composants» premier stade, c'est-à-dire une poignée de DIPs dans une boîte et quelques centaines de pages de littérature, absolument rien d'autre ! On vous laisse juges... Tous ceux qui l'ont essayé nous ont décrit en grand détail les difficultés rencontrées à l'oscilloscope pour arriver simplement à *faire tourner* le kit. Ceux qui ont l'habitude de l'oscillo double trace apprécieront des injonctions telles que «régler l'horloge à 490 nanosecondes en haut et 490 nanosecondes en bas» (à peu près infaisable). Puis le problème d'extension mémoire, pratiquement impossible, d'un bus non bufferisé...
- le même, second stade. Il y a en plus un circuit imprimé dans la boîte, combinant processeur, un peu de mémoire et E/S. C'est mieux, mais il faut ajouter alimentation, voyants, clés, pupitre, coffret, connecteurs... et surtout on est *obligé* d'utiliser des périphériques classiques d'ordinateurs, à interface RS-232 C par exemple, et on sait ce que ça coûte au total quand on a fait l'addition du tout ! Quant à l'aspect et à la portabilité, mieux vaut ne pas en parler : c'est le plat de spaghettis qui revient (notez que nous n'avons rien contre ces malheureux spaghettis en tant que plat chaud et fumant, à la carbonara c'est délicieux, en électronique, c'est autre chose...),
- **le kit système**, c'est celui d'Alcyane. Une série de périphériques non conventionnels sont spécialement développés, de façon à s'extraire autant que possible du domaine des périphériques classiques d'ordinateur jugés beaucoup trop chers. L'idée de base est d'utiliser autant que faire se peut des matériels éprouvés de grande diffusion, dont le rapport performances/coût est élevé à cause de la fabrication en série : téléviseurs du commerce, magnétophones, etc. Si certains périphériques sont acceptables tels quels, on les interface de façon à réduire les coûts le plus possible (disques souples, imprimantes).

Le kit est alors complet jusqu'à la dernière vis, la dernière entretoise : aucune mise au point à faire (nous l'avons faite pour vous) il n'y a qu'à assembler et tout marche à la mise sous tension.

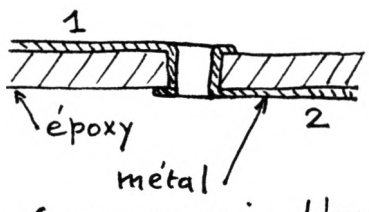
Enfin, le soft langages, écrit en assembleur pour être le plus compact et efficace, est livré complètement adapté aux périphériques du *système.*

4 - 1 - 2. Les étapes

Trois versions successives de la machine furent réalisées. La première, entièrement câblée en fil, tourna à notre grand contentement été 75 : quelle émotion lorsque le μP exécuta ses premières instructions vues en jolis créneaux verts à l'oscillo ! Intel annonça alors la disponibilité de nouveaux circuits, baissa les prix de certains autres, et nous jetâmes au panier des éléments de design douloureusement développés, telle notre horloge en TTL qui donnait de si beaux créneaux Ø 1 et Ø 2..., bien plus beaux, nous devons le dire, que ceux du 8224 qui l'a remplacée, mais qui marche O.K. quand même ! Le Sicob 75 approchait et nous décidâmes de transcrire le nouveau design directement en circuits imprimés. Le long travail commença alors, divisé pour chaque circuit en plusieurs phases :



Les plots en pointillé sont de l'autre côté de la carte

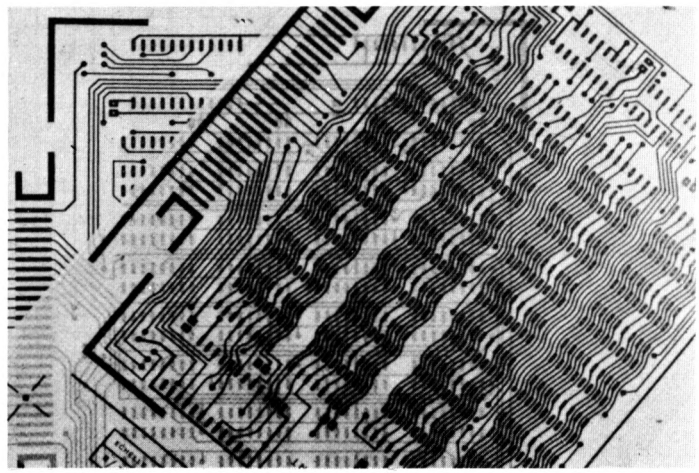


Coupe grossie d'un trou métallisé

— *l'implantation :* c'est le royaume de la combinatoire. Une fois le schéma théorique au point, on dispose physiquement les DIP (dessin sur calque à l'échelle 2) en recherchant la meilleure disposition pour la liaison de leurs pattes conformément au schéma de principe (théorique).

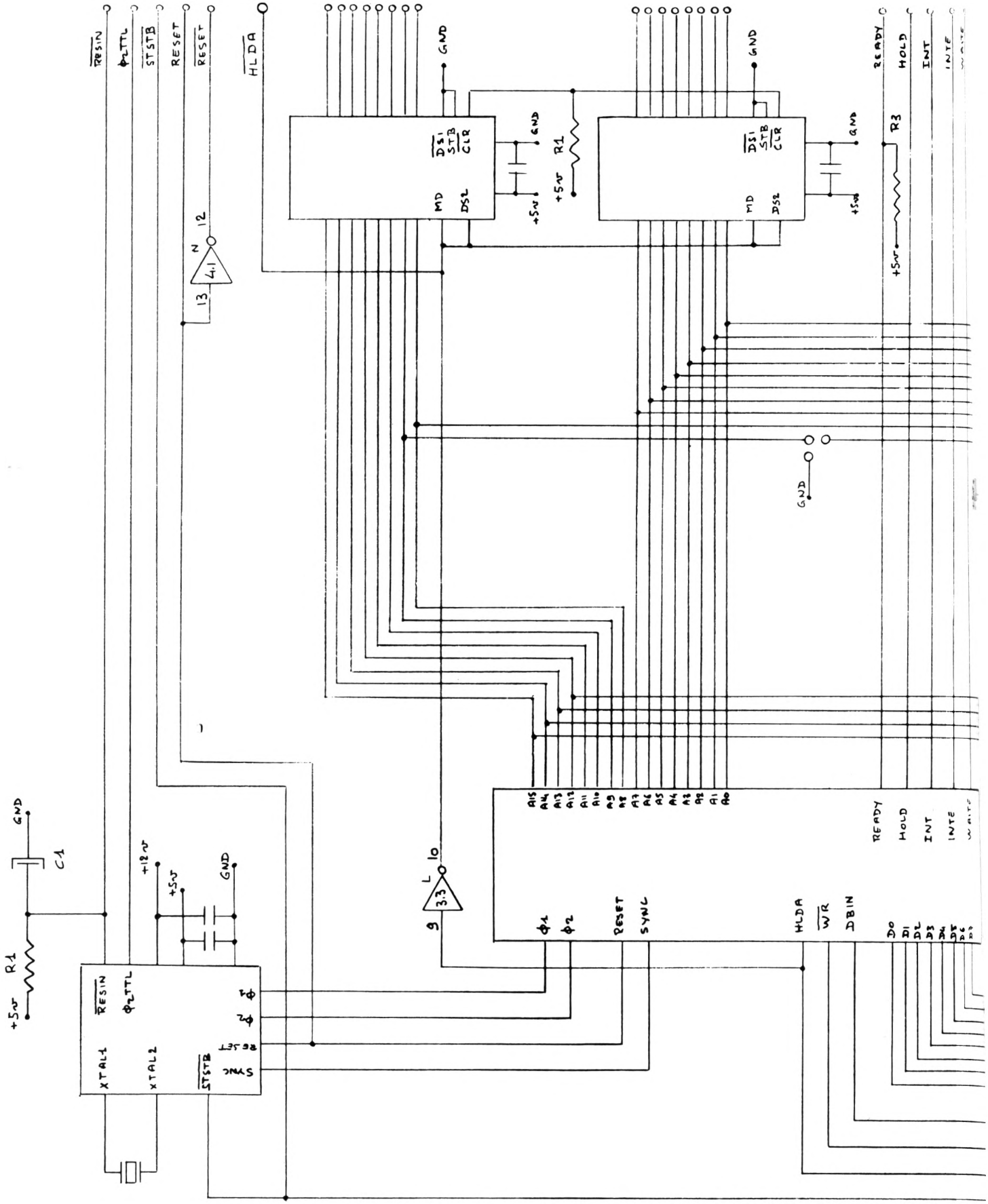
Les circuits imprimés sont nécessairement double face, vu le nombre élevé de points à relier entre eux par les pistes de cuivre étamé. La méthode générale consiste à orienter en gros les pistes à angle droit sur les deux faces comme l'indique le croquis, les trous métallisés permettant de relier entre elles les pistes conductrices des faces opposées.

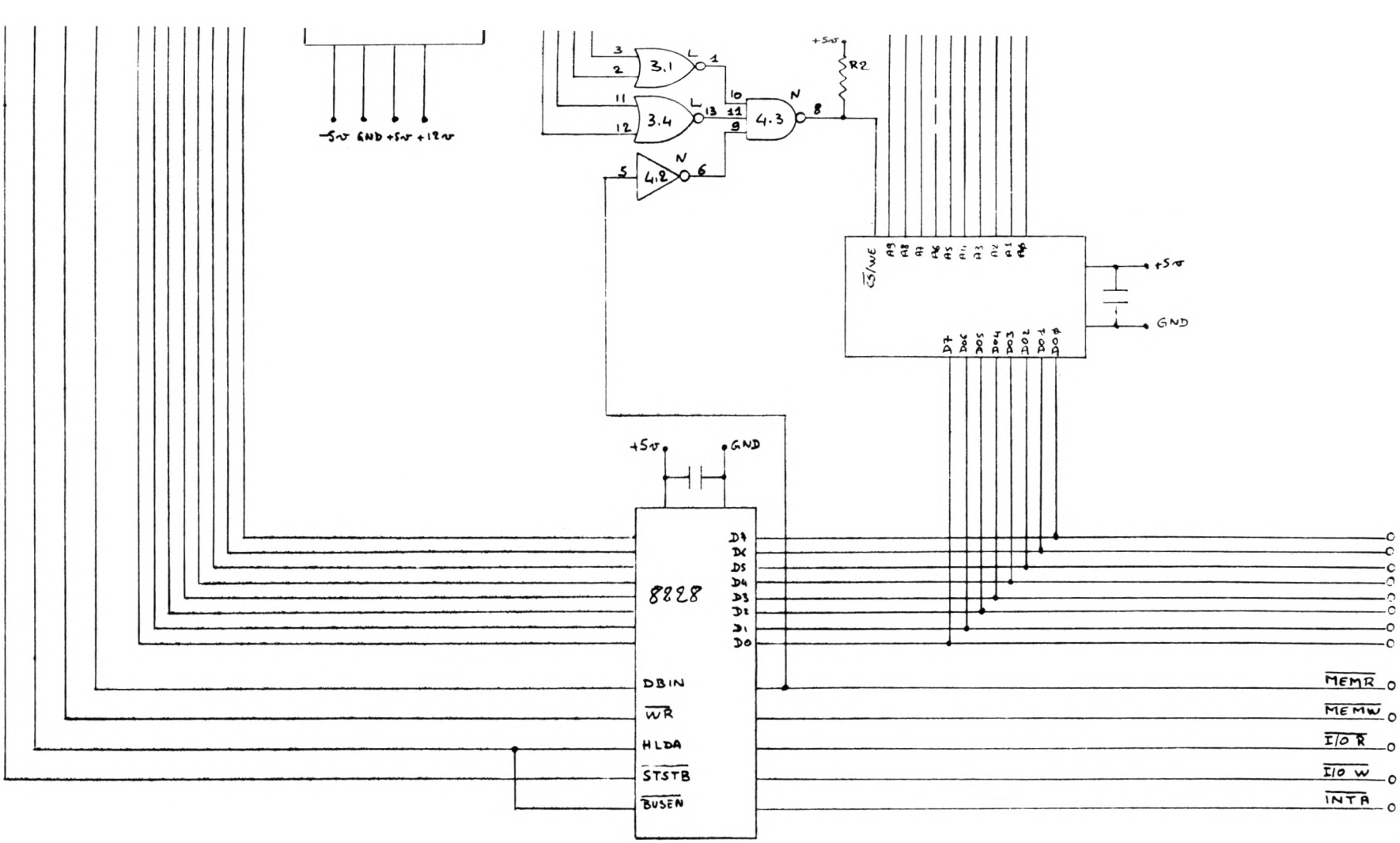
Ce travail est un casse-tête chinois de la plus belle eau, qu'on peut également considérer — ce fut le cas ! — comme un bon passe temps en soirées prolongées et en week-end...



— *les mylars :* une fois l'implantation jugée satisfaisante, après contrôle soigneux on la transforme en masque de fabrication des circuits imprimés. Compte tenu des outillages de fabrication existants, tout ce travail (implantation et masques) se fait sur une grille qui est une affreuse combinaison de système anglo-saxon et de système métrique : elle est en dixièmes de pouce ! Tout est donc au pas de 2,54 mm (et sous-pas de 1,27 mm) qu'on retrouve d'ailleurs des pistolets 6,35 aux moteurs caoutchouc des modèles réduits et aux mitrailleuses lourdes de 12,7...

Un travail méticuleux au ruban adhésif calibré et aux pastilles auto-collantes transférées sur un support transparent de mylar (présentant peu de variations de dimension sous l'influence des différences d'humidité dans l'air) permet de matérialiser le dessin d'implantation qui devient *le mylar...* appellation courante confondant le support et le travail terminé. Il y a deux mylars, un pour chaque face de circuit,


RESIN
Φ2TTL
STSTB
RESET
RESET
HLDA
GND
DS1
STB
CLR
MD
DS2
+5v
R1
R3
READY
HOLD
INT
INTE
4.1
N
12
13
C1
+12v
L
3.3
10
9
Φ1
Φ2
RESET
SYNC
HLDA
WR
DBIN
D0
D1
D2
D3
D4
D5
D6
D7
XTAL1
XTAL2
STSTB
SYNC



+12
+5
GND
-5

MOD_CPU_8080_1001
JUILLET 75

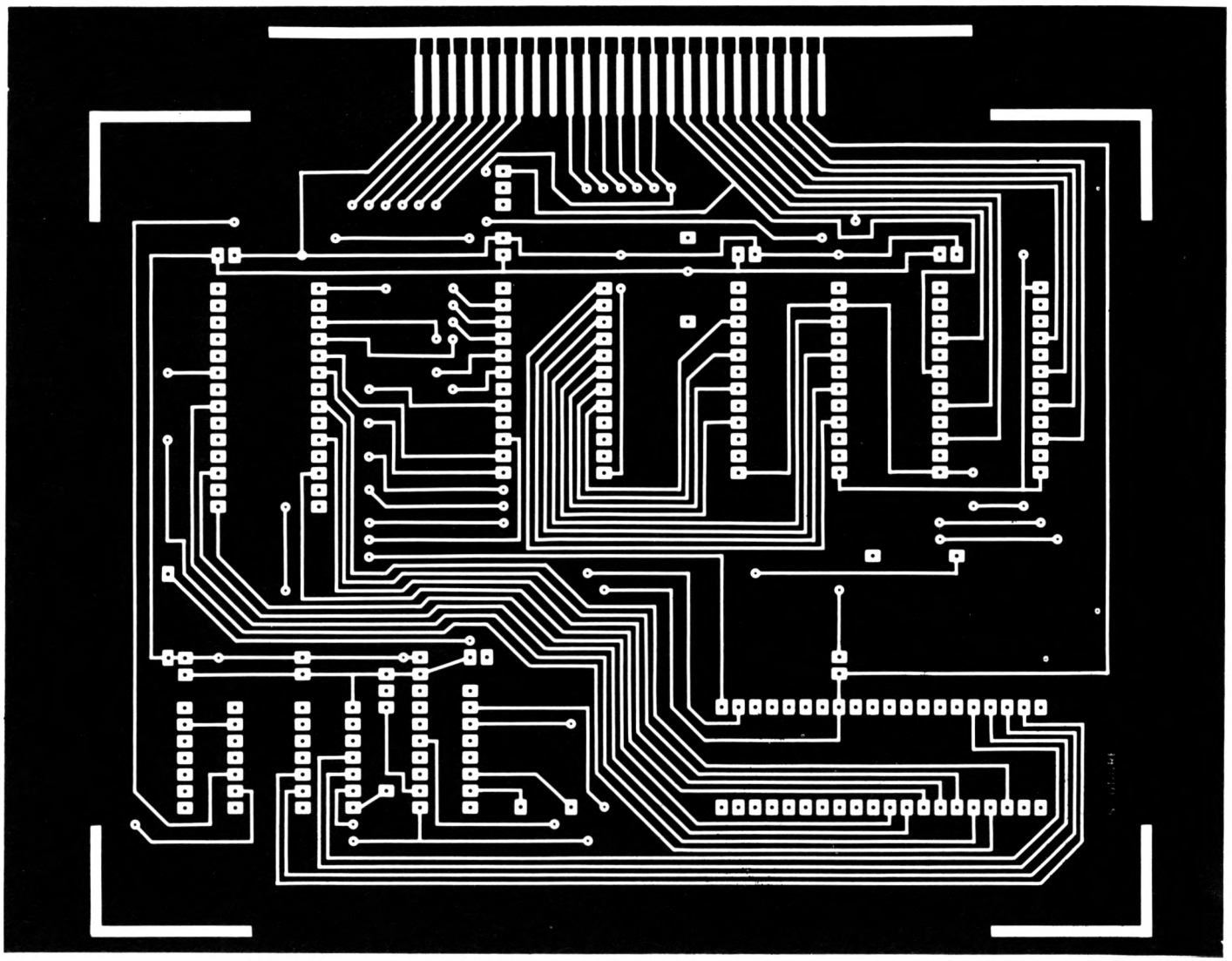
45

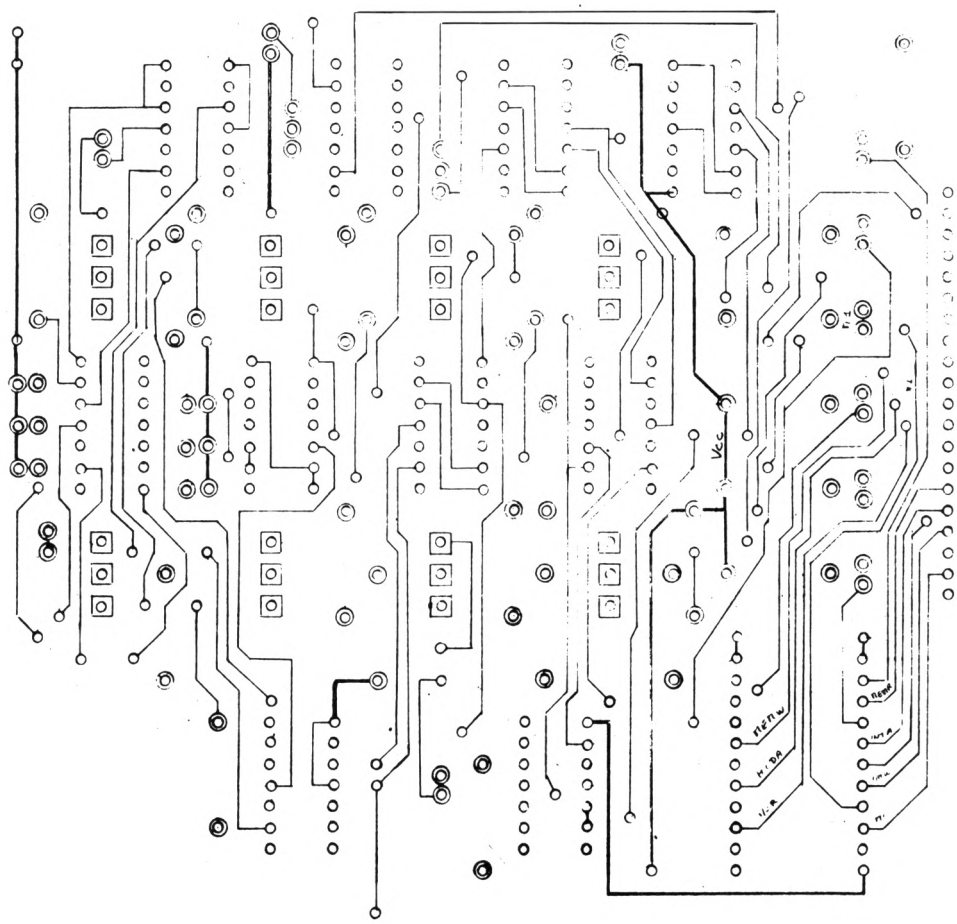
— *le croquis d'exécution* indiquant cotes extérieures et repères du circuit imprimé,

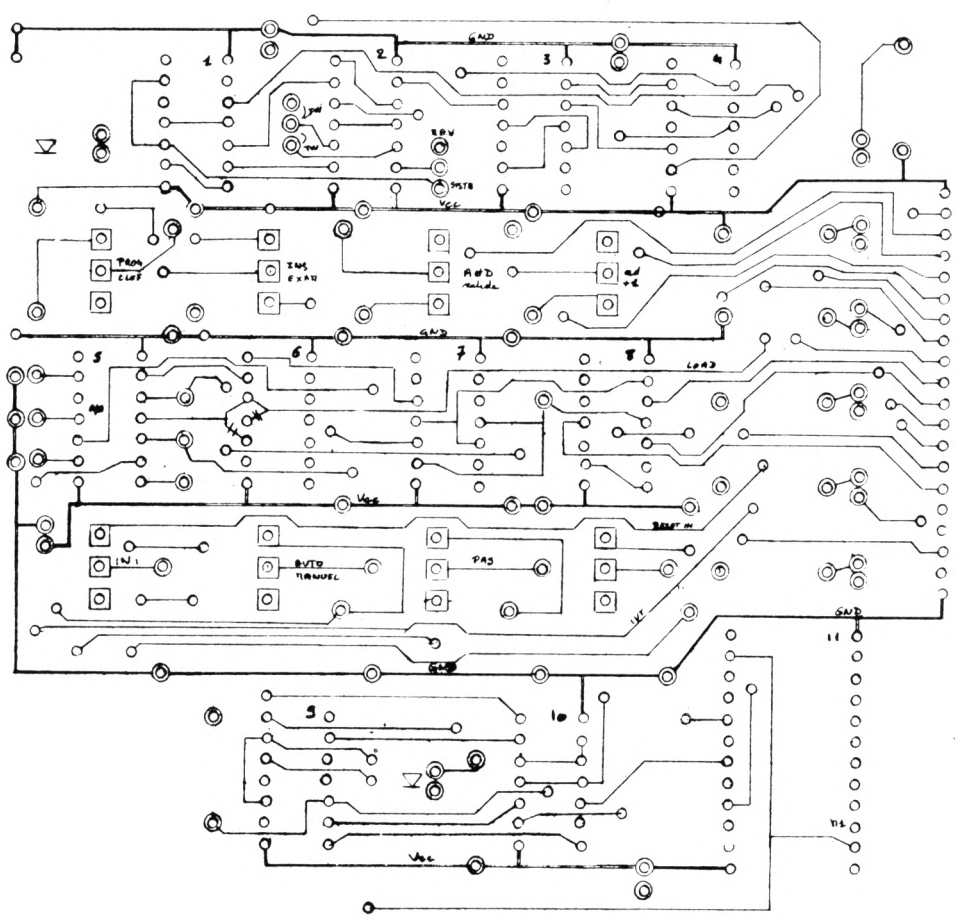
— *le masque :* il est réalisé par les fabricants de circuits imprimés qui photographient nos mylars en les réduisant à l'échelle 1 : 1 définitive. Le choix d'un bon fabricant est difficile, car des circuits imprimés avec 1 000 trous métallisés ou plus exigent la plus extrême qualité de réalisation.

— *les circuits imprimés :* quelques semaines plus tard on les reçoit, brillant dans leur enveloppe de papier cristal, soigneusement étamés, recuits, nettoyés aux solvants, fabriqués dans le meilleur matériau, le verre-époxy translucide vert insensible à l'humidité, très dur, avec des languettes de contact dorées à l'or fin, percés d'innombrables trous, recouverts d'une dentelle miroitante, nous ne pouvons nous empêcher de les trouver beaux...

Comme prévu, (c'est la même chose qu'en programmation) il y eut un bug par-ci, un bug par-là : c'était attendu, et nous avions pris le risque sciemment pour gagner du temps en évitant le long travail de câblage d'un deuxième proto (dont la retranscription d'ailleurs s'accompagne elle aussi de bugs variés !). Le debugging à l'oscillo fut très rapide, et se traduisit par quelques coupures de pistes par grattage et quelques fils soudés sur les circuits imprimés : à vrai dire, pas tellement, pas plus que sur les circuits imprimés des matériels *de série* de beaucoup de constructeurs ! La deuxième version d'Alcyane, qui était donc la machine de pré-série, tourna début 76 et nous la mîmes en test des journées entières, travaillant sans relâche en mémoire dans tous les octets et tous les bits, pour évaluer la fiabilité qui se révéla impeccable, supérieure à celle des minis auxquels nous avions accès. Depuis, d'ailleurs, les rapports de fiabilité officiels de Intel confirment sur les 100 000 premiers 8080 livrés, totalisant 31 200 000



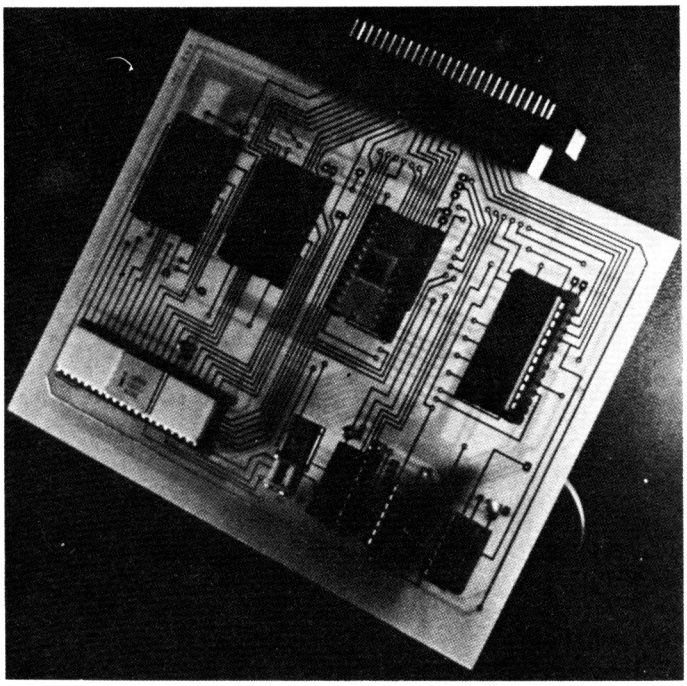

Vcc


GND
1
2
3
4
RAV
SYSTB
Vcc
PROG CLOF
INS EXAM
A&D
GND
5
6
7
8
LOAD
Vcc
RESET IN
IN1
AUTO MANUEL
PAS
GND
11
GND
9
10
Vcc

heures de fonctionnement, le chiffre incroyablement bas de 8 pannes seulement.

4 - 2. STATIQUE OU DYNAMIQUE ?

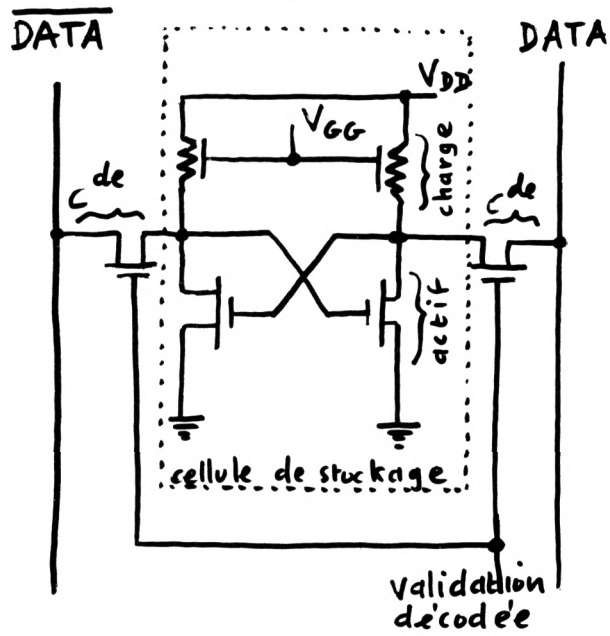
L'une des décisions les plus délicates dans le début de la conception avait été le choix de la technologie mémoire : bien sûr, il n'était plus question des ferrites, relégués au musée des antiquités, mais des mémoires en circuits intégrés existant en deux types très différents :



Circuit imprimé processeur de l'un des prototypes

— les mémoires statiques : elles utilisent comme cellule de stockage pour chaque bit une bascule à six transistors, dont seule une moitié est conductrice à un instant donné. Si c'est la première on a un 1, si c'est la seconde, on a un 0. Cette technologie est d'une fiabilité à toute épreuve. Par contre, elle utilise une cellule de stockage assez encombrante et consommant pas mal de courant. Les DIP disponibles stockent 1 024 bits,

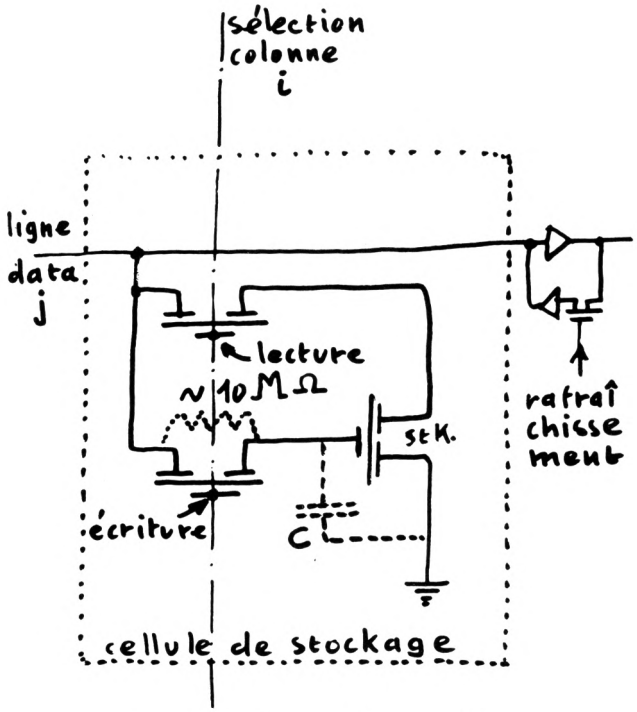
— les mémoires dynamiques : ce sont des DIP de 4 096 bits, à faible consommation, qui utilisent comme cellule de stockage la capacité (parasite) entre électrodes d'un transistor. Cette capacité sera chargée pour un bit 1 et vide pour un bit 0. Mais elle fuit, et se décharge : il faut la *rafraîchir* impérativement toutes les quelques millisecondes. Il s'agit là d'une servitude sérieuse, entraînant une bonne quantité d'électronique, mais surtout très gênante en présence de DMA et d'interruptions.



stockage d'un bit dans une mémoire statique (les deux transistors du haut servent de résistance de charge limitant le courant)

Nous subîmes d'abord l'attrait de la grande capacité des dynamiques, mais quelques faits isolés nous défrisaient. Par exemple, alors qu'avec les statiques la plupart des constructeurs avaient jugé inutile le bit de parité, ce qui était un bon progrès par rapport aux ferrites (encore que sur les énormes 7090 à ferrites utilisées pour la mission Apollo (à raison de quatre en parallèle !), il n'y en ait pas : de toutes façons, s'il y a une erreur de parité, le programme qui est en version compilée se met en bombe instantanément, qu'il y ait détection de parité ou non), avec les dynamiques on vit apparaître la série Eclipse de Data General avec 5 bits de parité pour chaque mot de 16 bits ! Des bruits couraient sur les «hoquets» périodiques de mémoires dynamiques... Bref, de plus en plus refroidis, nous décidâmes de jouer la prudence, et de choisir la statique, quitte à passer pour rétrogrades.

Un an plus tard, on l'a dit, Alcyane tournait comme une montre. A ce même moment, le seul fabricant américain de micros en kit était obligé de rappeler toutes ses cartes mémoires en circulation : elles étaient dynamiques... Entre temps les 4 K bit dynamiques s'étaient révélés comme les circuits intégrés jugés les plus difficiles à produire par l'ensemble des fabricants de composants, et



Mémoire dynamique à rafraîchissement par colonne (l'information est lue par le transistor du haut, amplifiée à droite et réinjectée dans stK à travers le transistor du bas). Le stockage est dans le "condensateur" C qui fuit à travers les ≈ 10 MΩ du transistor bloqué en 5 à 10 ms.

d'aucun prétendaient même qu'elles présentaient de subtils effets de température...

Les cartes I K et 4 K octets d'Alcyane sont donc des cartes statiques à structure toute simple, et l'alimentation électrique a été largement dimensionnée pour les supporter, avec des transfos à enroulements imprégnés qui sont de vrais tracteurs.

4 - 3. LA SERIE !

En février les mylars reçurent les changements correspondant aux fils, et en mars la troisième version d'Alcyane, était prête. Dès les tests réalisés, sans la moindre surprise, la fabrication fut lancée et la première série de kits livrée.

Le plus en retard... ce fut la documentation ! Sa page liminaire était la suivante :

Note aux premiers utilisateurs d'Alcyane

«Les premiers utilisateurs d'Alcyane — ceux qui nous ont fait confiance dès le Sicob 75 — sont pour nous plus que des clients ordinaires, que les milliers qui viendront par la suite. A ce titre, ils ont droit à des avantages... et des inconvénients. Ils ont dû attendre quelques semaines supplémentaires leurs machines que nous avons voulu impeccables dès le départ, et aujourd'hui nous venons leur demander un service.

La rédaction de la documentation pour un ordinateur en kit est une tâche toute nouvelle, compliquée pour le vaste éventail de ses utilisateurs : cette documentation, nous le sentons, sera difficile à mettre au point à cause de cette diversité. C'est la raison pour laquelle nous ne l'avons pas mise directement sous presse dès le départ : ce sont vos commentaires, vos critiques, sur lesquels nous comptons, qui nous permettront de la faire après réalisation des modifications que vous aurez souhaitées.

Naturellement, nous serons à votre disposition pendant cette période transitoire pour répondre oralement ou par écrit à vos questions, et vous aider ainsi dans toute la mesure du possible pour le montage et l'utilisation d'Alcyane.

Dans la suite, le club des utilisateurs commencera à fonctionner. Il deviendra très certainement un carrefour d'échange de programmes et d'expériences particulièrement variés, enrichissant continuellement l'environnement d'Alcyane... et donc la vôtre».

D'astucieuses suggestions surgirent de toute part pour faciliter le montage, qui de façon générale fut jugé simple à faire, et la documentation s'enrichit de jour en jour. En particulier, ceux qui avaient déjà monté un poste de radio ou un ampli trouvèrent que les soudures étaient infiniment plus rapides sur les circuits imprimés, ceci étant dû à la faible inertie thermique des pistes et des pattes des DIP, à l'étamage et au nettoyage poussé des cartes. Les coups de téléphone de ceux qui pour la première fois avaient donné vie à un ordinateur commencèrent, et nous reçumes en particulier un faire-part si sympathique que nous ne résistons pas à l'envie de le reproduire ici !

Ne vous effrayez pas trop des avertissements sur l'électricité statique et les MOS : un de nos jeunes amis américains a été mis dans un tel état d'inquiétude par des instructions de montage en provenance des USA qu'il a soudé dans l'herbe sur le campus de HEC, pieds nus et le dos appuyé sur un tuyau en zinc ! En réalité, aucun problème n'a été rencontré en suivant nos conseils simples.

Et maintenant, au fer à souder !

Philippe LECARPENTIER a le plaisir de vous faire connaître la naissance

d'ALCYANE

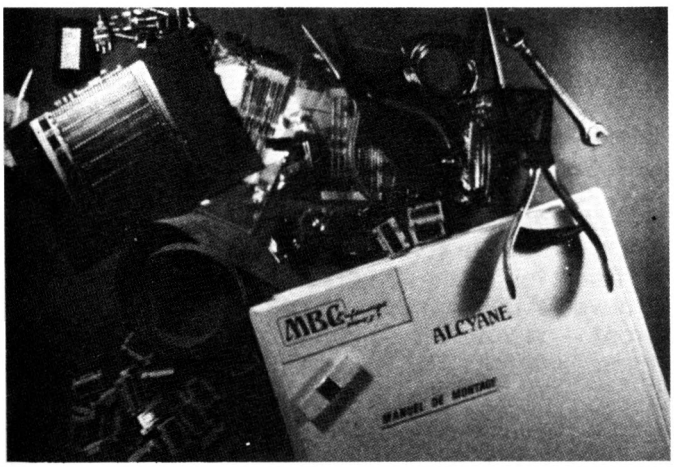
survenue le jour de Pâques 1976.

La gestation fut un peu longue (on l'attendait dès Noël !) mais l'enfant promet : après quelques vagissements lumineux désordonnés, elle a sur l'heure accompli ses premiers pas (de programme).

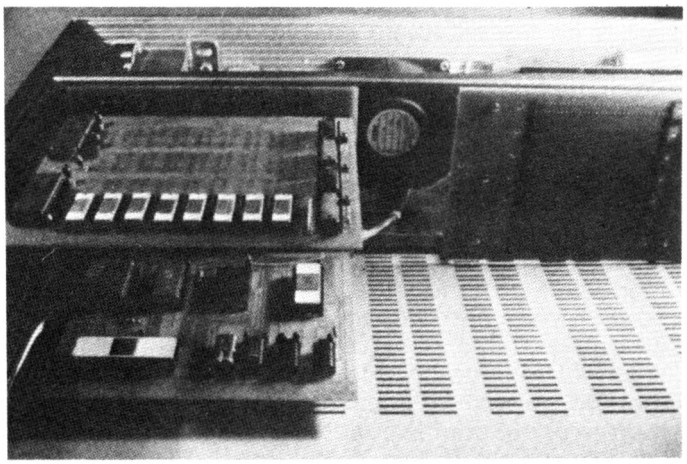
CHAPITRE 5
PROBLEMES PRATIQUES

Cette fois, le parcours n'est plus de 18 mois, il est de deux week-ends seulement, mais bien remplis.

Pour ceux qui n'ont pas vu l'intérieur d'une machine montée, les clichés photographiques indiquent la disposition générale à l'intérieur du coffret. Pour le reste, si vous savez reconnaître par quel bout il faut tenir le fer à souder... vous êtes prêt à commencer !



Les éléments du kit sur la table de travail : le circuit imprimé qui brille est la *carte fond de* panier, autrement dit le bus.



Disposition générale dans le coffret : les cartes des modules sont horizontales et s'enfichent dans les 14 connecteurs des cartes fond de panier verticales.

5 - 1. MANIPULATION DES DIP MOS

Le colis contenant votre Alcyane en pièces détachées vient d'arriver : nous espérons vivement que votre premier réflexe sera de parcourir ce guide, et non d'appliquer le proverbe anglais «When all trials have failed, read instructions...» (26)

Nous allons résumer en quelques mots les points les plus importants à observer pour le montage de votre kit, points qui sont développés plus loin en détail :

— les circuits intégrés MOS, pratiquement inusables une fois montés, peuvent par contre être claqués par manipulation si l'opérateur est chargé d'électricité statique : lisez en détail le paragraphe correspondant, vous éviterez tout ennui ;
— étudiez soigneusement tout le chapitre 5 avant de faire des soudures : la gaffe est de souder un composant du mauvais côté d'un circuit imprimé. Vous direz «il faut le faire, vu le repérage» : c'est vrai, mais reconnaissez bien le terrain avant de faire le premier pas, nos planches sont particulièrement complètes et sont là pour ça ;
— certains composants simples tels que condensateurs, diodes électro-luminescentes, ont deux connexions dont la polarité est à respecter ;
— enfin, dans le coffret, les cartes doivent être placées composants **sur le dessus** quand on les insère dans les connecteurs.

Dans l'ensemble, le montage du kit d'Alcyane est facile, car l'emploi des circuits intégrés à très haute intégration les plus récents de Intel (en particulier les 8212, 8228 et 8224) réduit considérablement le nombre de composants à souder. Le câblage, presque intégralement assuré par les circuits imprimés (que nous appellerons «cartes») comporte un minimum de fils. Encore les écheveaux de fils habituels sont-ils ici remplacés par des câbles plats, rendus possibles par la disposition des pistes conductrices sur les cartes spécialement destinées à cet effet.

Par simple examen vous vous rendrez compte que, n'étant pas conçue par le hobby, tout dans Alcyane est de première qualité : c'est une machine étudiée pour un usage intensif sans défaillance, dans le commerce, l'enseignement, la médecine ou l'industrie. Les cartes, coûteuses, sont en verre époxy épais double face avec pistes en cuivre étamé et recuit. Les composants MOS viennent tous de chez Intel, le leader mondial en matière de microprocesseurs et de circuits mémoire, réputé pour la qualité de ses fabrications et de ses contrôles. Les composants TTL viennent de chez National Semiconducteur, NS, également fabricant d'excellente renommée. Le transformateur d'alimentation est à enroulements imprégnés. Le ventiteur a une durée de vie espérée supérieure à cinq ans en service continu. Nous avons mis tout notre cœur dans la conception de cette machine, car nous l'avons faite après tout pour notre plaisir, et — nous en sommes sûrs — également pour le vôtre.

Nous appelons DIL ou DIP (dual in-line packages, boîtiers doubles en ligne) ces caramels rectangulaires à deux rangées de pattes, noirs, gris ou blancs, que sont les circuits intégrés MOS et TTL utilisés dans Alcyane. Tous les DIL de NS sont du TTL, et se manipulent par conséquent sans précautions particulières (voir le paragraphe Identification des composants pour reconnaître les TTL et MOS).

Par contre, les transistors, à effet de champ, qui au nombre de quelques douzaines jusqu'à 5000 sont contenus dans les DIP MOS, ne supportent pas des tensions élevées sans détérioration : ceci veut dire que l'électricité statique peut être un problème. On la rencontre lorsque l'atmosphère est sèche (donc en particulier en hiver quand le chauffage fonctionne), spécialement lorsque le revêtement de sol est en fibres synthétiques. Dans ce cas, le fait de marcher dans une pièce ou de se lever de son siège provoque dans le corps une accumulation d'électricité statique qui se décharge, parfois même sous forme d'étincelles, à travers les objets qu'on touche.

Des vêtements en nylon ou autre fibre synthétique produisent également de l'électricité statique, ou même la simple action de se coiffer.

A titre de précaution, le microprocesseur en particulier est piqué dans une mousse conductrice spéciale qui court-circuite les décharges. Ne l'en détachez pas avant l'instant de l'utiliser, et ne le manipulez pas plus que nécessaire. Heureusement, depuis quelques temps les circuits MOS sont fabriqués avec une protection, incorporée sur les connexions de grille, constituée d'un réseau de diodes et de résistances : *tous les circuits MOS d'Alcyane sont ainsi protégés.*

Dans les deux paragraphes qui suivent, vous trouverez des conseils précis qui vous permettront — si vous les suivez à la lettre — *de ne rien avoir à craindre de l'électricité statique.*

5 - 2. NOTE SUR LE SOUDAGE ET LES OUTILS

Pour mener à bien la construction de ce kit, il faut un petit fer à souder, une main ferme (ou bien

(26) Quand tous les essais ont échoué, lire les instructions...

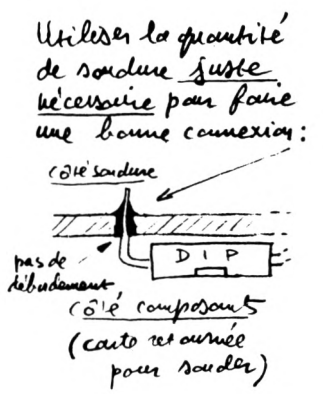
appuyée !) et éventuellement une grande loupe à pied (selon votre vue...) : suivez soigneusement les instructions, et vous ne rencontrerez aucun problème.

Le fer à souder, de type «stylo», doit avoir une panne de diamètre 1,5 mm au plus, et une puissance de 20 watts maximum, s'il n'est pas régulé. Nous vous recommandons soit un SEM 712, soit un Weller régulé en température, soit un fer Antex type X ou CN avec panne de 3/64e de pouce. Il faut mettre la panne à la terre, en y tortillant le bout d'un fil électrique fixé sur un robinet de radiateur par exemple, et vérifier que le fer correspond bien à la tension du secteur utilisé pour éviter que la panne soit trop chaude ou trop froide.

Les composants sont à enficher sur les cartes aux emplacements prévus, leurs pattes étant à souder au revers de la carte. Après mûre réflexion, nous n'avons pas utilisé de supports dans le kit : les composants seront donc soudés directement, comme dans la plupart des matériels professionnels, et ceci pour des raisons de fiabilité. Nous avons en effet rencontré des problèmes de mauvais contacts avec des supports, et comme par ailleurs la fiabilité a été poussée au maximum dans la conception et la sélection des composants, il aurait été dommage de conserver des supports.

Utilisez la meilleure soudure possible : alliage 60/40, diamètre 1 mm, décapant neutre garanti non corrosif.

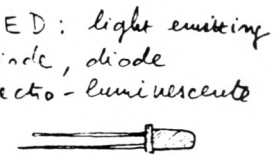
Les circuits intégrés et autres composants modernes ne sont pas aussi facilement endommagés par la chaleur qu'on le croît communément, et aucune précaution particulière à ce sujet n'est nécessaire. Cependant, faites chaque soudure aussi vite que possible, en amenant simultanément au contact le fer et la soudure sur l'extrémité de la patte de DIL à souder : n'amenez pas la soudure toute fondue sur le bout de la panne, car le décapant s'évaporerait trop tôt. La durée



normale pour faire une soudure est de l'ordre de deux secondes. Ne rien déplacer entre l'instant où on a retiré fer et soudure, et celui où la soudure se solidifie. Elle doit être lisse et brillante, non granuleuse ou en cristaux. Nettoyer fréquemment la panne comme dit ci-après. Vous trouverez dans le kit un morceau de tresse à dessouder : c'est une tresse métallique imprégnée d'un flux spécial, qu'on utilise pour retirer un excès de soudure (court-circuitant deux pattes de DIP par exemple).

Pour cela, bien nettoyer la panne du fer (un chiffon de fil en 16 épaisseurs, ou une serviette de restaurant en papier sont parfaits, et on ne se brûle pas) pour enlever tout reste de soudure, mettre le bout de la tresse sur la soudure à défaire et poser la panne dessus. La mèche va pomper la soudure. Couper ensuite l'extrémité de la tresse imbibée de soudure. On peut aussi, sans utiliser de tresse, retourner la carte, appliquer le fer et la secouer pour faire tomber la soudure fondue.

Nous attirons votre attention sur le fait qu'il est difficile sans équipement spécial de retirer d'un circuit imprimé un composant qui a été soudé.



Si vous vous apercevez d'une erreur, renvoyez-nous la carte pour dessoudage. Si par exemple, une LED est à remplacer, ou à changer de polarité, coupez ses pattes à 5 ou 6 mm au-dessus de la carte, et soudez dessus les pattes de la diode neuve préalablement coupées à la longueur.

A part le fer, les outils nécessaires sont :

— une pince coupante (on peut utiliser un coupe-ongles) pour couper le fil et les pattes des résistances, condensateurs, etc.
— une pince plate à longs becs demi-ronds, utile pour plier les pattes des résistances, condensateurs, etc.
— une pince à dénuder, *indispensable* pour dénuder le bout des fils des câbles plats (le modèle le plus simple, non automatique, suffit),
— des ciseaux ordinaires pour couper le câble plat.

5 - 3. PLAN DE TRAVAIL ET PRECAUTIONS

Pour éviter les effets de l'électricité statique, nous vous recommandons de recouvrir votre plan de travail d'une feuille d'aluminium ménager. Avant de commencer à souder, touchez la feuille avec le bout de la panne du fer, et posez-y votre doigt en même temps. Refaites-le de temps en temps pendant l'assemblage, et en particulier si vous vous êtes levé pendant le travail.
Cette précaution vous assurera que toutes les

charges statiques sont éliminées : elles se seront toutes écoulées à travers la mise à la terre de la panne du fer. On peut également mettre la feuille d'aluminium à la terre, et y relier le fer par un fil fin. Avec un Weller ce n'est pas nécessaire. Lavez-vous les mains avant de travailler : poussières ou traces grasses sur les cartes rendent les soudures difficiles à faire et peu fiables.

Vérifiez et revérifiez toutes les positions de composants sur une carte avant de commencer à souder.

Inspectez soigneusement chaque carte après assemblage. Assurez-vous que les soudures sont lisses et brillantes. Vérifiez qu'il n'y a pas de ponts de soudure entre points rapprochés, éclaboussures ou filets de soudure, parfois dissimulés sous la résine.

Vérifiez les deux côtés de la carte, bien que les soudures ne soient faites que d'un côté.

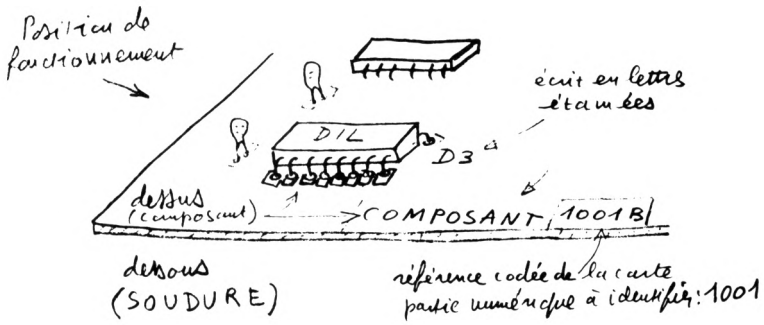
Ne vous pressez pas trop de finir : c'est alors que vous ferez des erreurs !

Ne soudez qu'aux endroits indiqués (pattes des composants et fils...). Les trous nombreux à étamage circulaire qui réalisent les connexions entre les pistes conductrices des deux faces des cartes sont métallisés à la fabrication, et n'ont pas besoin de recevoir de soudure.

5 - 4. CONSEILS D'ASSEMBLAGE

Ces notes n'indiquent pas exactement le meilleur ordre de montage : elles sont surtout pour but d'expliquer les séquences de montage indiquées plus loin afin de les alléger.

L'essentiel d'Alcyane est constitué de cartes (circuits imprimés) : sur un côté sont placés les



composants, et les soudures sont à faire de l'autre. En position de fonctionnement dans la machine, la face COMPOSANT est placée **sur le dessus**. Pour réaliser les soudures, il faut bien sûr poser les cartes à plat à l'envers.

Le travail doit procéder carte par carte. Chacune d'elles a *un nom, et une référence :* ne tenez compte *que de sa partie numérique.* S'y rapportent :

— un sachet contenant les composants à y souder. N'ouvrez pas tous les sachets, vous auriez à tout re-trier en risquant des erreurs : n'ouvrez que le sachet de la carte en cours. Une fiche collée dessus liste les composants,

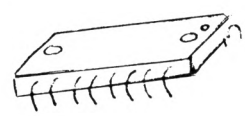
— une planche, vue du côté COMPOSANT, indiquant :
- l'emplacement de chaque composant, repéré par un code (reproduit par étamage lorsque la place sur la carte était suffisante pour l'y faire figurer),
- l'orientation de chaque composant : attention !
- le repérage des connexions de sortie de la carte,
- le schéma interne des DIL avec le rôle de leurs broches : ceci a représenté un travail important (ce genre de renseignement ne figure normalement jamais dans un kit) que nous avons entrepris pour tous ceux qui s'intéressent au fonctionnement de la machine (enseignement, connexions spéciales, dépannage 2e échelon au niveau composants, voir manuel de maintenance et d'interfaçage),
- une liste des composants avec leur code et leur désignation commerciale (pour les DIL) ou un descriptif (pour les composants discrets : valeur des résistances, condensateurs, types d'interrupteurs, etc.) et repérage de polarité,
- des indications de variations possibles de câblage (matérialisation d'adresse et carte mémoire, cycles d'attente, tout ceci expliqué plus loin),

— un schéma de principe (qui ne sert pas pour le montage d'Alcyane, même usage que le schéma interne des DIL), fourni seulement dans le manuel de maintenance et d'interfaçage.

La première étape du travail de montage d'une carte consiste à *identifier* les composants du sachet :

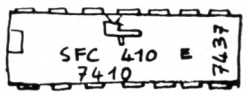
a) *les circuits intégrés en boîtier DIL :* code D sur liste des planches.

Ils sont rectangulaires, longs de 2 à 5 cm. Les DIP à conditionnement plastique sont noirs ou bruns. Ceux à conditionnement céramique sont blancs avec un capot central métallique. Le nombre de pattes va de 14 (2x7) à 40 (2x20) en passant par 16, 20, 28, etc.



Type : il y a en général deux nombres sur le dessus de chaque DIL, l'un correspondant à son type, l'autre à son numéro de série de fabrication : ce

dernier comprend les deux derniers chiffres de l'année suivis du numéro de semaine en deux chiffres. Malheureusement, les DIL TTL les plus utilisés appartiennent à la famille 7400 dont les types sont 7414, 7402, 7474, 7400, 74107, 74161, etc. ce qui donne une affreuse confusion pour les DIL fabriqués en 1974... Ci-dessous, un DIL monté sur une carte processeur Alcyane de présérie : la liste de composants de la carte processeur permet de voir qu'elle reçoit un 7402 et un 7410, c'est donc de ce

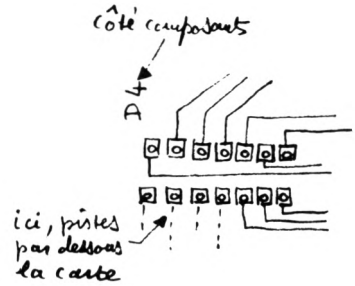


Le bon code est normalement dans le sens de la longueur et au voisinage immédiat du nom du fabricant.

dernier qu'il s'agit. Le comptage des pattes donne une vérification supplémentaire. Ne pas tenir compte des préfixes ou suffixes alphabétiques (P = plastique, C = céramique, et autres), en général en tête du n° de série.

Chez Intel, deux codifications existent : le premier des 4 chiffres peut être *indifféremment 3 ou 8* (3212 = 8212).

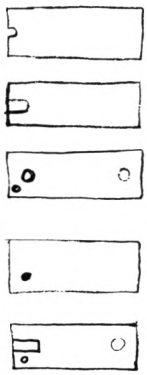
Orientation : sur les cartes, les emplacements des DIP sont très reconnaissables : deux rangées parallèles de trous entourés de plots à souder rectangulaires.



En règle générale, les trous métallisés de passage d'une face à l'autre sont entourés d'un étamage circulaire (ne pas souder), et les emplacements à souder, eux, d'un étamage rectangulaire.

La symétrie des DIP donne deux positions possibles (4 si en plus on se trompe de côté de la carte !), et une seule est bonne, bien entendu. Le repérage se fait par un repère sur le boîtier du DIP, repère qui peut avoir plusieurs aspects différents :

— une encoche semi-circulaire à un bout,
— une dépression de même style,
— une dépression circulaire un peu plus profonde qu'à l'autre bout et accompagnée d'un point en creux,
— une petite bosse,



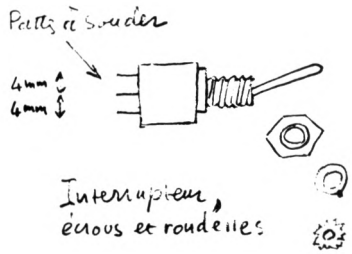
Dans cette position, les codes se lisent à l'endroit

— une dépression comme le cas n° 2 mais avec un point (et une faible marque circulaire à l'autre bout).

Ce repère est systématiquement figuré sur nos planches par un demi-cercle au bout, et il faut *impérativement* l'identifier.

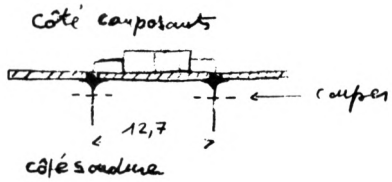
b) *les interrupteurs :* code SW (switch) sur liste (on les appelle également *clefs,* sauf le SW secteur).

Leurs pattes de sortie sont pleines pour soudure sur les cartes, et avec un œil pour soudure sur fils : il n'y a que l'interrupteur secteur de ce type dans Alcyane. Les 32 autres se décomposent en :



— 27 à deux positions stables (bistables) : on peut les mettre indifféremment dans les deux positions possibles,
— 5 à rappel central à ressort : les placer de façon à ce qu'on comprime le ressort quand on appuie de haut en bas sur le levier, la carte (celle de gauche de la face avant) étant dans sa position verticale (interrupteurs TRAM, TRA + 1, RAZ, PAS et INT),

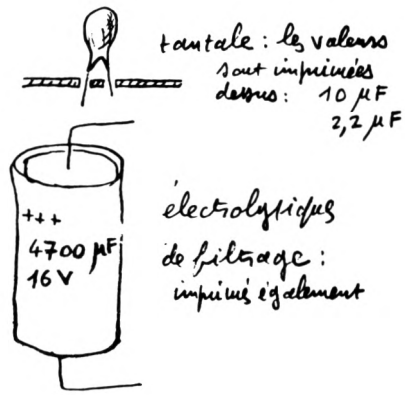
c) *les résistances :* code R sur liste.
Leur plier les pattes à la pince, ou en réalisant un petit gabarit si l'on est perfectionniste.

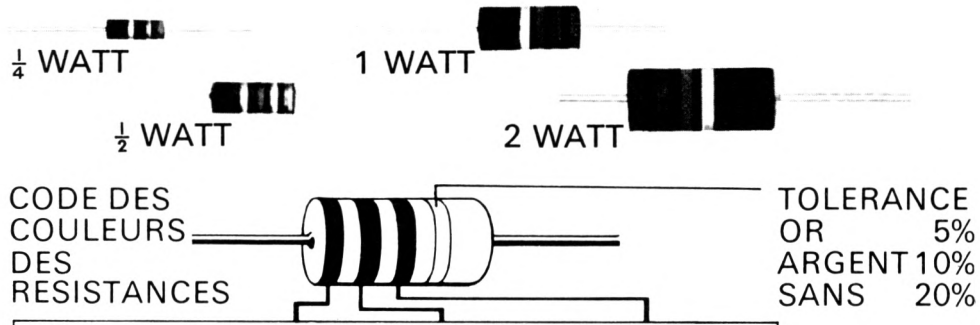


Toutes les distances entre trous sur les cartes sont des multiples de 1/10e de pouce (2,54 mm), et pour les résistances nous avons normalisé à 5 pas. On peut placer les résistances dans n'importe laquelle des deux positions possibles, mais il est plus net d'aligner les couleurs s'il y en a plusieurs (face avant).
Les valeurs des résistances sont indiquées par les bandes de couleur qui les entourent, selon le code indiqué ci-après.

d) *les condensateurs :* code C sur liste.
Vous en trouverez de deux catégories dans le kit :

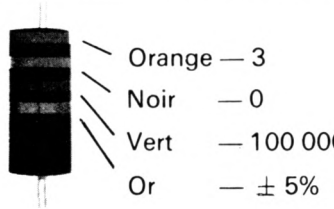
— petits condensateurs au tantale, à souder sur les diverses cartes (processeur, mémoire, etc.). Les positionner légèrement au-dessus de la carte comme indiqué,





COULEUR	CHIFFRE 1	CHIFFRE 2	MULTIPLICATEUR
Noir	0	0	1
Marron	1	1	10
Rouge	2	2	100
Orange	3	3	1,000
Jaune	4	4	10,000
Vert	5	5	100,000
Bleu	6	6	1,000,000
Violet	7	7	10,000,000
Gris	8	8	100,000,000
Blanc	9	9	1,000,000,000
Or	–	–	.1
Argent	–	–	.01

15 1 000
– 15 000
(ou 15 KΩ)

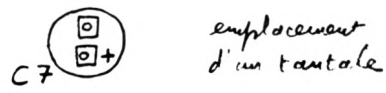


30 × 100 000 =
3 000 000
(ou 3 MΩ)

Le tableau est accompagné de deux exemples de calcul de valeur des résistances.

— gros condensateurs cylindriques, utilisés dans l'alimentation.

Les condensateurs ont souvent une polarité, c'est-à-dire que le fil + doit être mis à l'emplacement marqué + du côté composant de la carte :

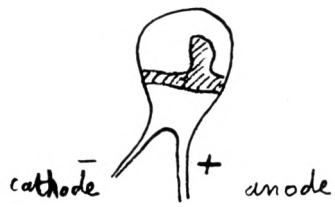


Pour repérer la polarité des tantales :

— le + y est parfois indiqué

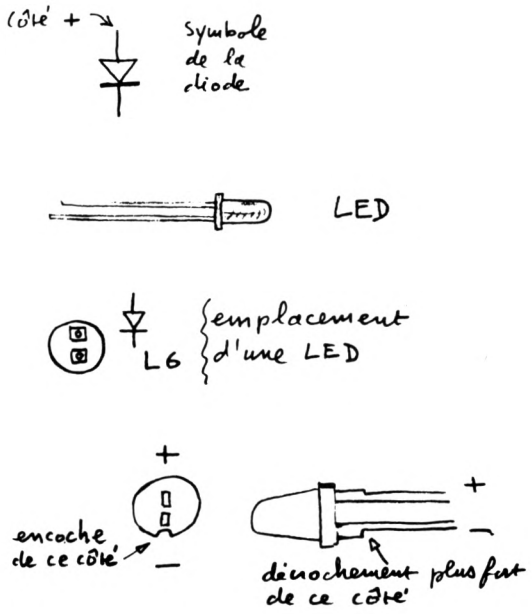
— sinon, tourner le condensateur de façon à avoir vers soi la tache de peinture surmontant la bague peinte de marquage (les pattes vers le bas) :

le + est alors à droite



e) *les diodes électro-luminescentes :* code L sur liste.

Les LED, dont nous avons dit un mot §3, ont également une polarité à respecter. Leurs emplacements sont marqués sur les cartes comme pour les condensateurs, mais accompagnés du symbole de la diode : la pointe du triangle indique le sens du passage du courant, qui va du + au —, et permet donc de repérer le côté + situé vers la base du triangle.



Le repérage des pattes de la LED est fait de la façon suivante : le côté négatif (*cathode*) est repéré par une petite *encoche*.

Par ailleurs, toutes les cathodes sont orientées *vers le bas* dans la machine (les LED sont sur des cartes verticales).

f) *Autres composants*

Sur les cartes de l'alimentation et les cartes interface, des composants différents des précédents sont utilisés. Leur variété étant grande, ils sont décrits au niveau des instructions de montage des sous-ensembles correspondants (micros-relais, potentiomètres, diodes de puissance, diodes ordinaires, régulateurs de tension, radiateurs, photo-coupleurs, etc.).

*
* *

Une fois l'étape d'identification des composants terminée, il restera à les placer sur la carte à leurs emplacements respectifs, avec la bonne orientation, puis à les souder après avoir retourné la carte : mais ceci n'est que le schéma général du travail, pour chaque carte il y a un ordre préférentiel à respecter, il est indiqué dans les instructions détaillées de montage de chacune d'elles.

5 - 5. DESCRIPTION GENERALE DE L'UNITE CENTRALE

L'unité centrale d'Alcyane est formée de six ensembles :

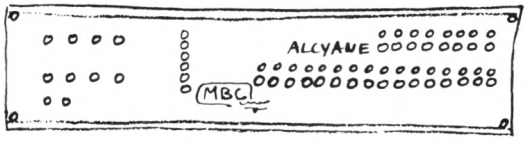
— le pupitre
— la carte fond de panier
— la carte processeur
— la (ou les) carte mémoire
— l'alimentation
— le coffret.

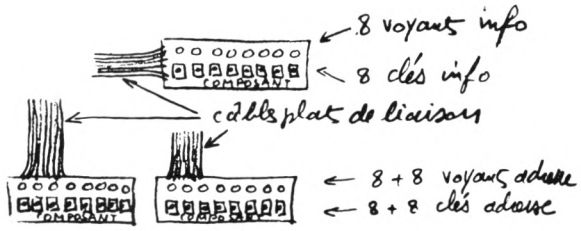
Nous allons les décrire très brièvement.

a) *Le pupitre*

C'est l'ensemble le plus complexe de la machine, mais il est formé de sous-ensembles simples au nombre de six, et c'est donc par lui que nous vous recommandons de commencer le montage :

— face avant : facile à reconnaître, 68 perçages, 4 vis fente en croix aux angles,
— 1 carte INFORMATION placée derrière la zone de même nom,

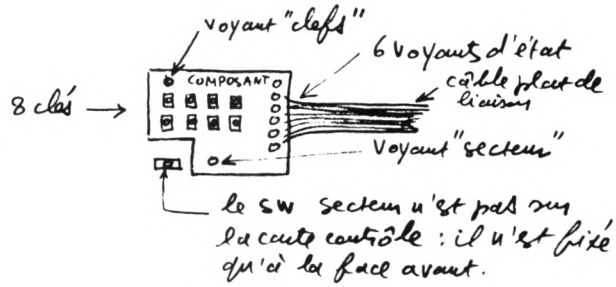




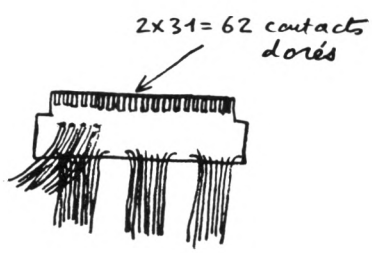
— 2 cartes ADRESSE identiques entre elles, placées derrière la zone adresse.

Chacune de ces 3 cartes porte 8 interrupteurs et 8 LED : elles se ressemblent beaucoup, bien les identifier (les deux dernières ne prennent leur personnalité de moitié gauche ou droite de l'adresse qu'au moment où on les relie à la carte de connexion du bus).

— 1 carte CONTROLE. Elle porte les 8 clés de fonctions dont le rôle est expliqué dans le Guide de manipulation du pupitre, ainsi que les voyants.



— 1 carte de LIAISON sur le bus, rassemblant les extrémités des 4 câbles plats provenant des circuits précédents.



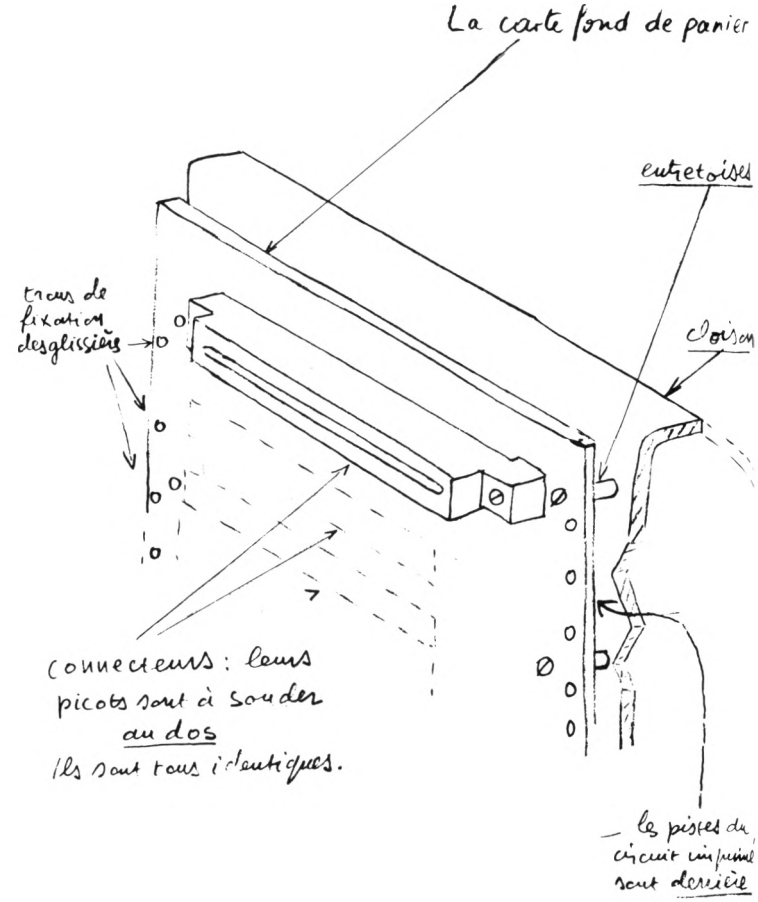
Cette carte est horizontale, toutes les autres sont verticales. Cette carte s'enfiche dans un connecteur de la carte FOND DE PANIER. On peut donc séparer facilement le pupitre du reste de l'unité centrale.

b) *la carte fond de panier,* verticale.

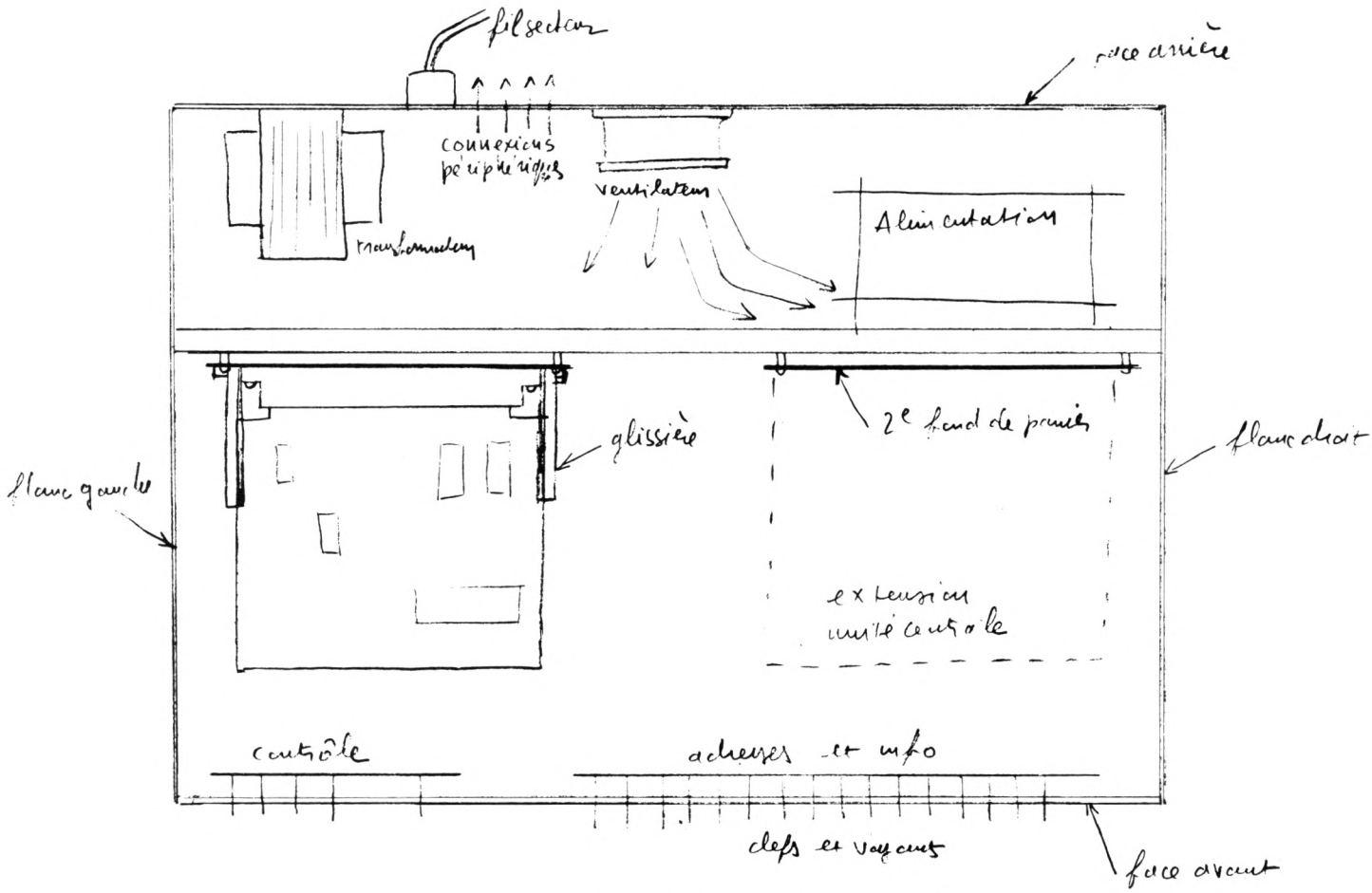
C'est elle qui matérialise le bus universel d'Alcyane, par les pistes situées à son dos, côté SOUDURE. Elle est fixée par vis et entretoises à la cloison centrale transversale qui relie les flancs gauche et droit du coffret.

Elle peut recevoir au total 7 connecteurs. Dans chacun d'eux, on enfiche une carte : carte processeur, carte mémoire, carte interface, carte de liaison face avant, etc.

Le principe du bus banalise les connecteurs, qui peuvent recevoir indifféremment n'importe quel type de carte dans n'importe quel ordre.



La cloison peut recevoir deux cartes fond de panier, qui sont alors reliées entre elles par câble plat à souder sur un 8e emplacement pour assurer la continuité du bus. Des glissières en plastique maintiennent les cartes par les bords, et sont fixées par vis et écrou près des bords verticaux de la carte fond de panier.



Vue de dessus de l'unité centrale

c) *la carte processeur*

Rien de particulier à signaler à ce niveau, sinon qu'elle comprend un emplacement dans lelquel il est possible de mettre une EROM (mémoire morte effaçable, erasable read-only memory) de la nouvelle série de Intel à haute intégration. Un choix entre deux types est possible :

— une 8704, capacité 4096 bits (512 octets),
— une 8708, capacité 8192 bits (1 K octet),

Ces EROM permettent de stocker en particulier un moniteur, ce qui évite d'avoir à inscrire manuellement un programme initial après chaque mise sous tension de la machine.

d) *la carte mémoire*

Il en existe de deux types : 1 K octets et 4 K octets. La première est prévue pour les petits systèmes ne comportant que l'unité centrale en général, la seconde est la carte mémoire normale de base pour les systèmes d'usage général.

Les DIP utilisés sont de type statique, et présentent par conséquent une grande sécurité de fonctionnement : chaque chiffre binaire est stocké dans une bascule, circuit à transistors pouvant prendre deux états stables. Dans l'un, la première branche du circuit est conductrice, ce qui représentera 1, dans l'autre, la seconde branche est conductrice, ce qui représentera 0. L'un ou l'autre de ces états peut persister indéfiniment tant que la mémoire est alimentée élecriquement.

Dans les DIP dynamiques, l'information est stockée sous la forme d'une charge électrique qui fuit progressivement, et qu'il faut rafraîchir toutes les quelques millisecondes. C'est une technologie d'utilisation assez délicate.

e) *L'alimentation*

Elle comprend :

— un transformateur de qualité, à enroulements imprégnés. Entrée 110 et 220 V alternatifs, sorties alternatives + 8V, + 16V et — 8V,
— trois sections de redressement, filtrations et régulation fournissant des tensions continues

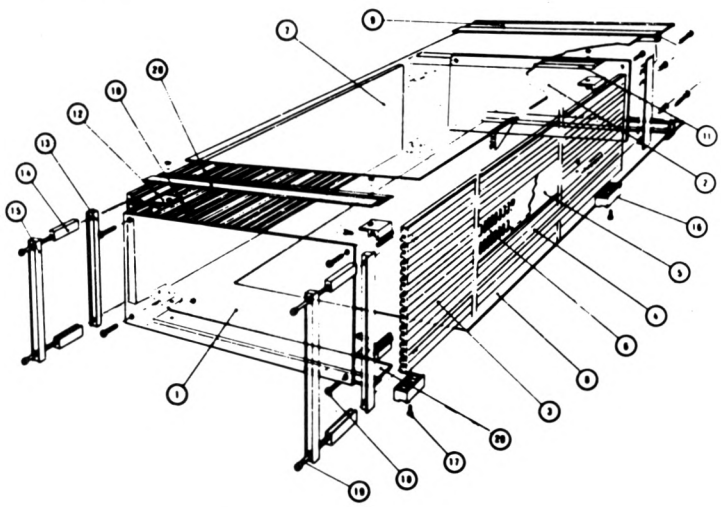
régulées avec précision à + 5V, + 12V et — 5V, nécessaires pour faire fonctionner le microprocesseur, la mémoire, le pupitre et les interfaces. En option : le — 12V,

— un ventilateur, recommandé en versions 1 K, nécessaire à partir de 4 K. Sa durée de vie est supérieure à 5 ans en service quotidien normal de 12 heures.

f) *le coffret*

C'est un coffret démontable en aluminium. La plaque du dessous est perforée pour laisser passer l'air de ventilation. On l'assemble à l'aide des vis fournies dans le kit. Les plaques du dessus et du dessous se glissent dans les rainures des flancs latéraux, et sont retenues par la face arrière et la face avant. Ces dernières sont fixées chacune par 4 vis. La cloison centrale transversale ajoute à la rigidité de l'ensemble.

Vue éclatée

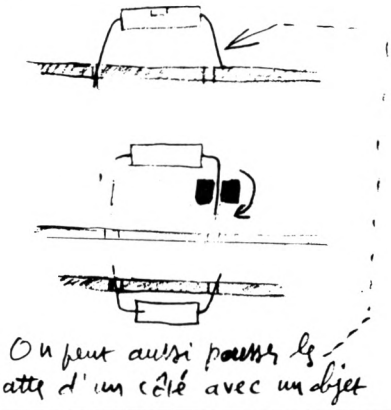


Sur le plan esthétique, nous vous signalons que de la feuille auto-collante plastique imitant le cuir noir (marque Alcor par exemple, dans les Prisunic ou autres) collée sur la plaque du dessus, améliore encore l'aspect général : mais ceci est affaire de goût personnel.

5 - 6. MONTAGE DE L'UNITE CENTRALE

a) *Carte INFORMATION : réf. 5000*

Après avoir identifié les composants, mettre les DIPs en place, bien orientés, du côté COMPOSANT de la carte. Les pattes d'un DIP sont normalement trop écartées pour rentrer dans les trous. Une première méthode consiste à les plier *légèrement* à la pince toutes ensemble d'un côté, en faire autant de l'autre, de façon à ce que le DIP puisse entrer, mais que les pattes restent encore assez écartées pour que le DIP tienne en place tout seul quand on retourne la carte pour le soudage.



Voici une autre méthode très valable, due à Francis Gernet, pour placer les DIPs sur les cartes :

POUR MONTER TRÈS RAPIDEMENT LES DIL 14 ET 16 BROCHES, SANS AVOIR A PLIER LES PATTES

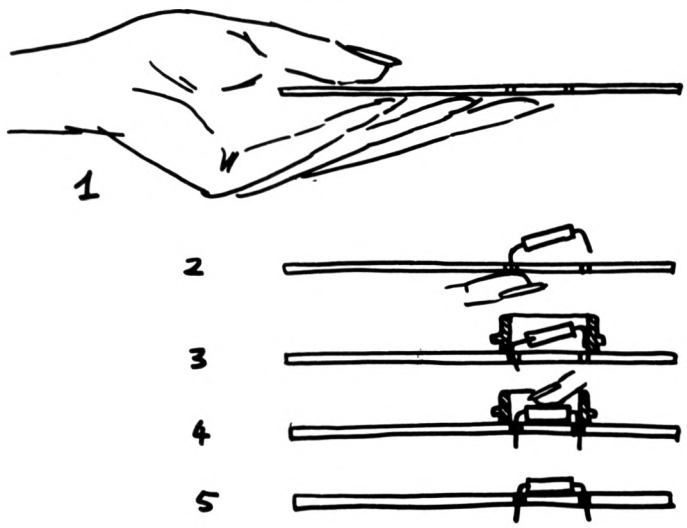
On prend le circuit avec la main gauche (bien propre !) en plaçant un doigt sous une des deux séries de trous (la marque « Composants » lisible vers le haut !).

Avec la main droite, on place les pattes d'un côté du DIL dans les trous (on se sera assuré que le DIL est dans le bon sens). Le DIL s'enfonce un peu mais les pattes ne ressortent pas de la plaque car elles sont bloquées avec un doigt de la main gauche.

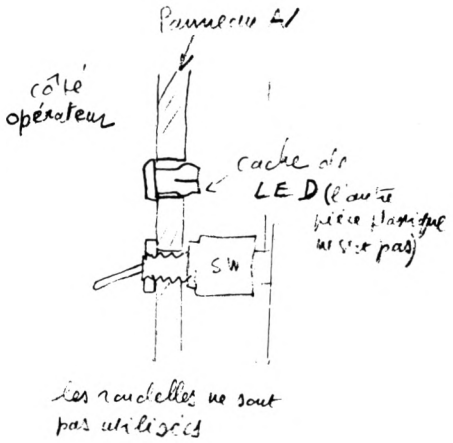
On prend avec la main droite (qui n'a plus besoin de tenir le DIL) un des petits containers à DIL (par exemple celui d'un 8216) et, partie débordante vers le bas, on en coiffe le DIL. La main gauche reste en position.

On appuie sur le DIL à l'intérieur du container, avec l'ongle, en libérant du doigt de la main gauche les pattes du DIL. Le container se retire de lui-même.

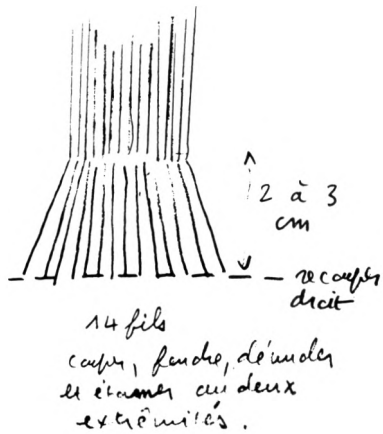
Le DIL est positionné rapidement (avec un peu d'exercice), sans action sur ses pattes, et il tient tout seul pour la soudure.



Vérifiez une nouvelle fois qu'il n'y a pas d'erreur, que les pattes sont bien enfoncées et que les DIP sont bien à plat, parallèles à la carte. Retournez la carte, posez-la à plat et soudez. Façonnez, montez et soudez ensuite les résistances puis les condensateurs, comme dit en 5-4 c) et d).

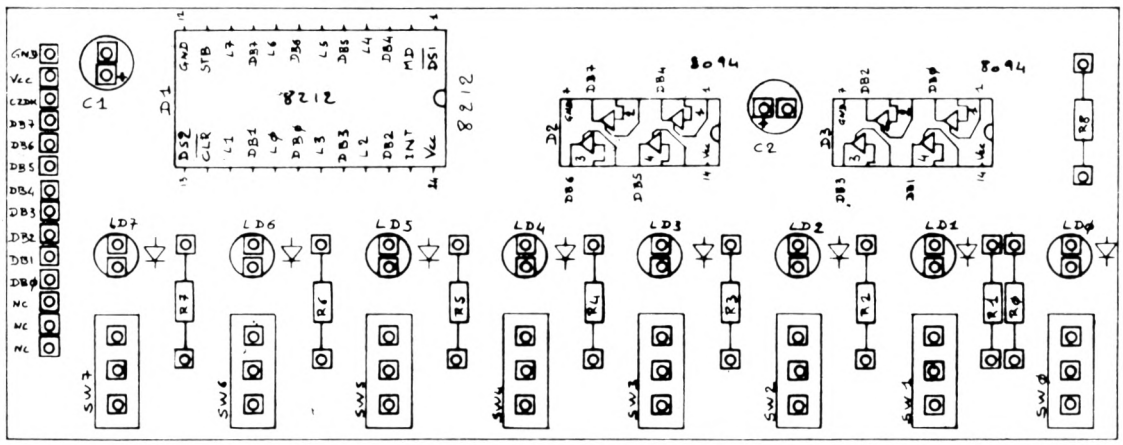


— identifier, placer et souder les DIPs puis les discrets,
— souder les interrupteurs après vissage sur panneau,
— les LEDs 34 cm pour l'adresse gauche
— le câble plat (AB 15 à AB8)
— la vérification 45 cm pour l'adresse droite (AB7 à AB0).



composants: R0 à R7 120 Ω
R8 1 KΩ
C1 C2 2,2 µf
LD0 à LD7 LED
SW0 à SW7 interrupteur 2 positions

D1 8212
D2 DM8094
D3 DM8094



COTE COMPOSANT REF 5000

b) *Cartes ADRESSE : réf. 5002 (deux cartes)*

Ces deux cartes sont identiques, le même montage est à effectuer pour les deux.

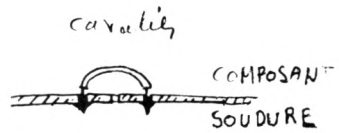
Procéder dans le même ordre que pour la carte INFORMATION :

c) *Carte CONTROLE : réf. 5001*

Bien que la carte soit plus chargée, et la sortie à 20 fils, le travail n'est pas différent et se fait dans le même ordre.

Attention aux orientations différentes des pattes + et — des condensateurs, elles sont très variées.

Sur cette carte, nous rencontrons pour la première fois des *cavaliers* (angl. strap) à souder aux emplacements S1 et S2 en fonction de diverses options (explications plus détaillées dans le manuel de maintenance et interfaçage). Un cavalier est un morceau de fil (utilisez le petit fil isolé fourni dans le kit) à couper, dénuder et étamer aux bouts, pour relier deux plots où on le soude, comme l'indique la figure. On peut aussi le dénuder complètement, c'est plus simple.



S1 : si le temps d'accès mémoire est supérieur à 580 ns (pour le 8080 A normal), ce qui est le cas des Alcyane standard premier trimestre 1976, relier b et c (a reste libre)

S2 : sauf instructions spéciales fournies avec votre machine, relier b et a (c reste libre).

d) *Carte LIAISON : réf. 6500*

Sur cette carte convergent les 4 câbles plats en provenance des 4 cartes précédentes. Ils sont à souder sur les 4 rangées de plots repérées FC1 à FC4 (flat connection) :

— FC1 : partie gauche de l'adresse, AB15 à AB8
— FC2 : partie droite de l'adresse, AB7 à AB0
— FC3 : information, DB7 à DB0
— FC4 : contrôle.

Cette correspondance est en outre facilement vérifiable que le nom des signaux. Les deux cartes ADRESSE, identiques, prennent leur personnalité de moitié droite ou gauche par leur longueur de câble et leur liaison l'une à FC1, l'autre à FC2.

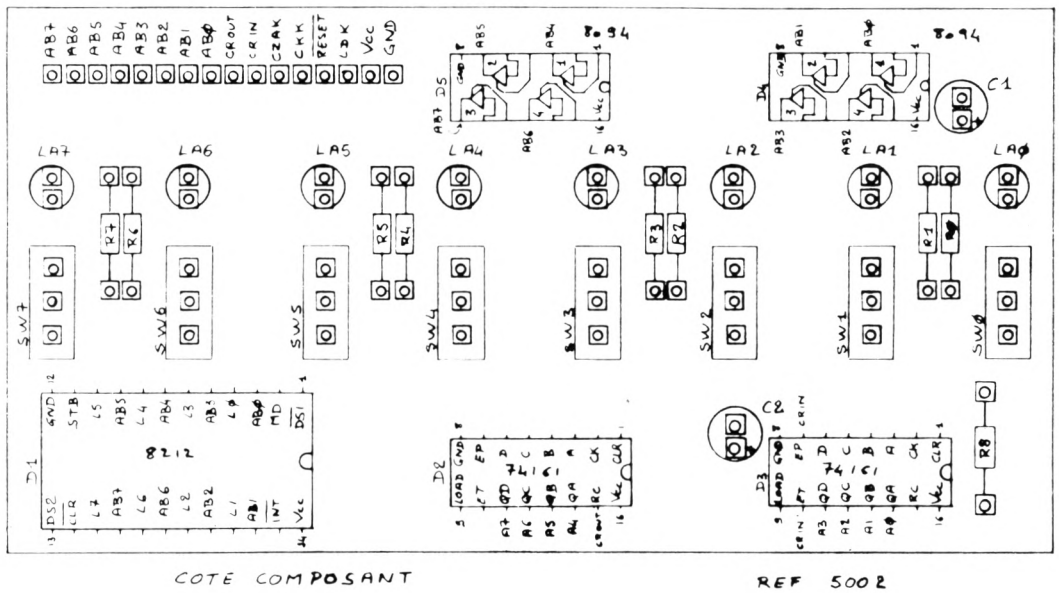
Consultez le tableau des cavaliers, et mettez-les en place en fonction des options particulières de votre machine. Pliez et soudez R1. Il n'y a aucun autre composant à souder sur la carte : les emplacements prévus correspondent à des DIPs pour des fonctions spéciales.

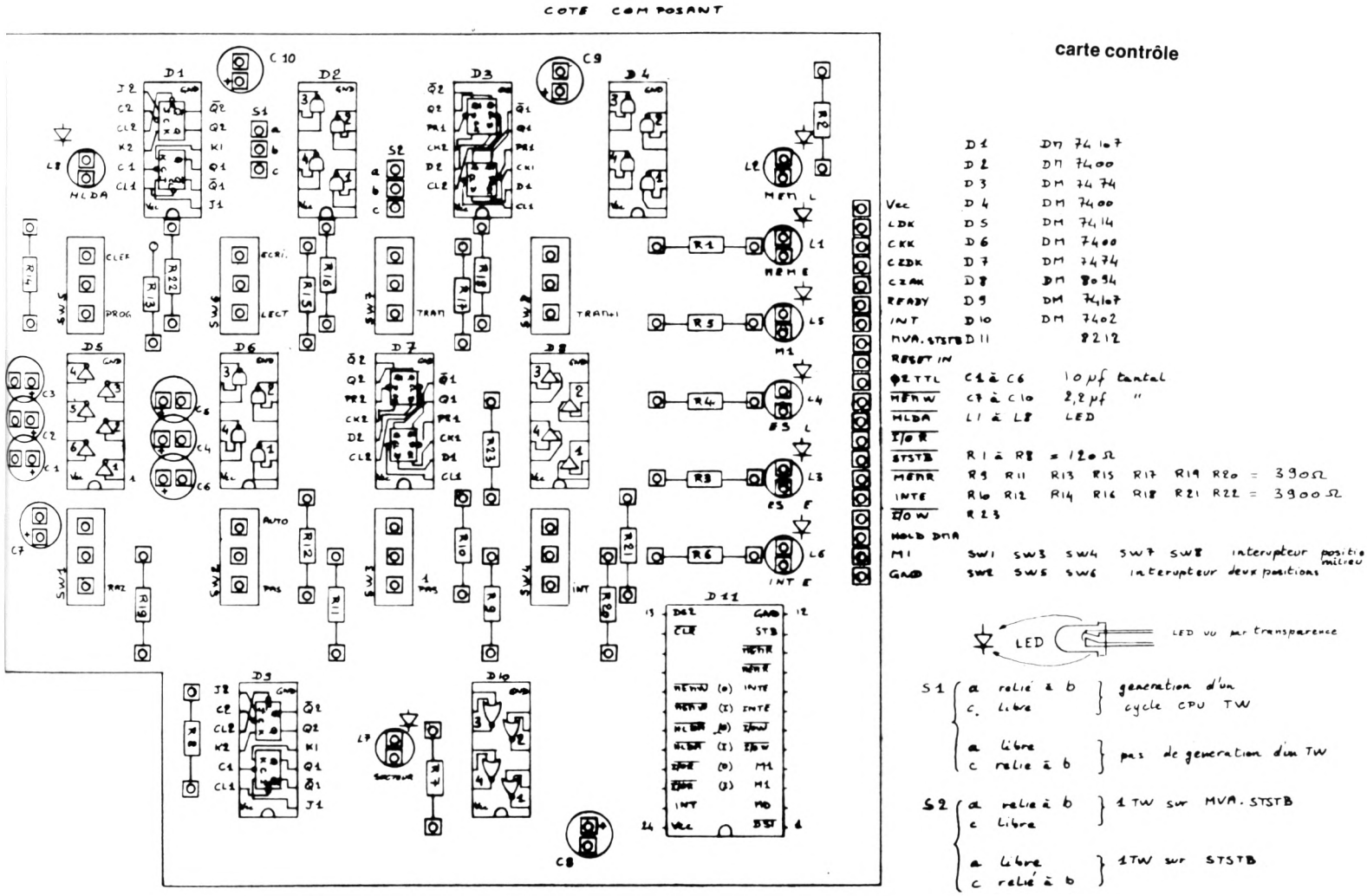
e) *Carte PROCESSEUR : réf. 1001*

Elle est facile à assembler. Montez d'abord les DIPs, en prenant des précautions particulières pour le microprocesseur MOS comme indiqué. Montez ensuite les discrets, en faisant attention

composant :	R0 à R7	120 Ω	D1		8212
	R8	1 KΩ	D2	DM	74161
	C1 et C2	2,2 µf goutte tantal	D3	DM	74161
	LA0 à LA7	LED	D4	DM	8094
	SW0 à SW7	interupteur 2 positions	D5	DM	8094

Carte adresse





aux polarités, puis le support du quartz en l'orientant comme sur le dessin, et chauffez longuement ses soudures. Le quartz peut y être enfiché dans n'importe lequel des deux sens possibles : il n'a pas de polarité. Il n'y a de cavaliers à souder que lorsqu'un DIP EROM 8704 ou 8708 (à monter sur support) est installé pour doter Alcyane d'un moniteur (voir tableau S1).

Les 8212 et 8228, qui sont en TTL Schottky, outre leur rôle de driver de bus, améliorent l'immunité au bruit et protègent les MOS : on peut donc manipuler la carte montée normalement et l'enficher dans son connecteur sans précautions spéciales (par principe, il vaut mieux tout de même prendre l'habitude de toucher la table ou le coffret en premier plutôt qu'une carte, et tenir cette dernière par les bords comme une plaque photo humide, ne serait-ce que pour ne pas graisser les contacts).

f) *Carte MEMOIRE 4 K statique : réf. 2001*

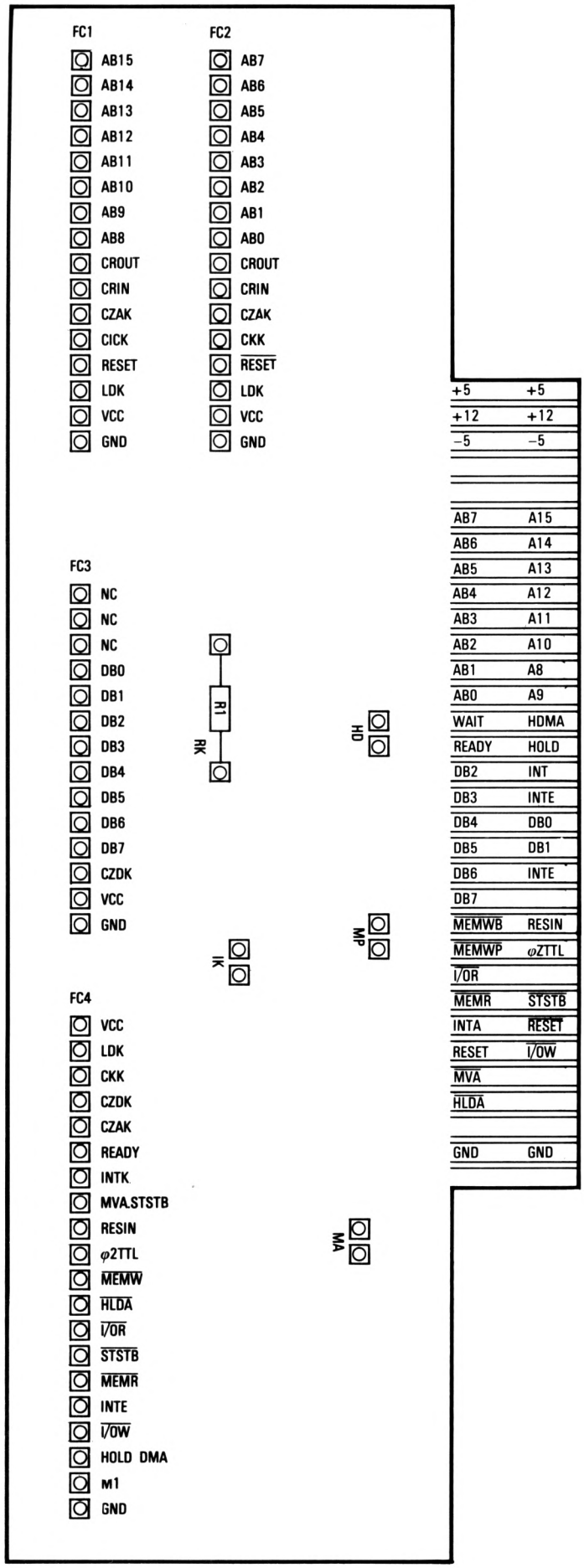
Aucun problème particulier pour cette carte, sinon les précautions de manipulation pour les 32 DIP MOS de la mémoire, et le nombre de soudures plus élevé que pour les autres cartes. Soudez assez rapidement, mais cependant sans vous presser.

Les cavaliers permettent de déterminer la zone des adresses de la carte mémoire comme suit :

S1 : si adresses entre 0 et 32 K { relier a et b / relier c et Vcc

si adresses entre 32 K et 64 K { relier a et c / relier b et GND

S2 : permet de choisir l'un des huit intervalles de 4 K à l'intérieur de la plage choisie par S1 :



Carte de liaison
Fave avant-bus

Intervalle	0-4 32-36	4-8 36-40	8-12 40-44	12-16 44-48	16-20 48-52	20-24 52-56	24-28 56-60	28-32 60-64
relier a et :	b	c	d	e	f	g	h	i

Tant qu'on n'a pas de moniteur, utiliser 0-4 K pour pouvoir se positionner en 0 par RAZ.

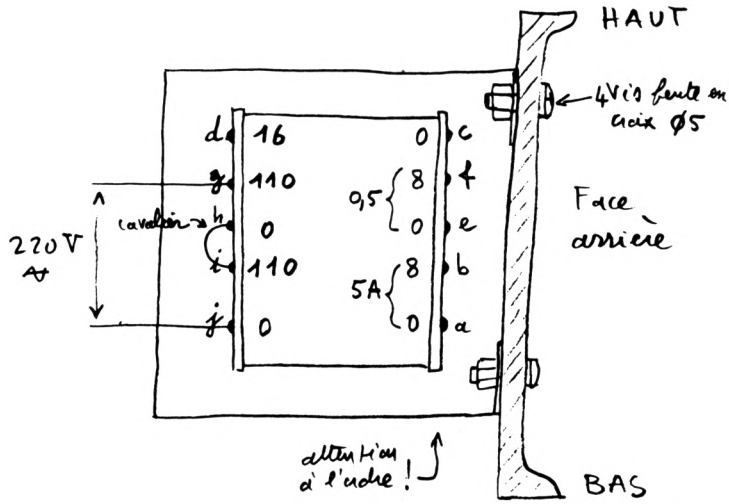
Quand on a le moniteur, changer le cavalier pour laisser l'espace 0-4 K disponible pour la mémoire EROM.

g) *Alimentation : réf. 7000*

L'alimentation comprend le transformateur et la carte réf. 7000 qui assure le redressement et la régulation du courant. Ces deux éléments sont fixés par vis et écrous sur la face arrière du coffret.

Repérage du transformateur

Deux séries de cosses sont à câbler : l'entrée (en 110 ou 220 V alternatif) et les sorties basse tension à relier à la carte 7000 (lire le paragraphe «Ventilateur» avant de câbler).



Les cosses du transfo sont repérées à l'encre sous vernis par les valeurs des tensions et intensités.

Le câblage est à réaliser partout en gros fil isolé.

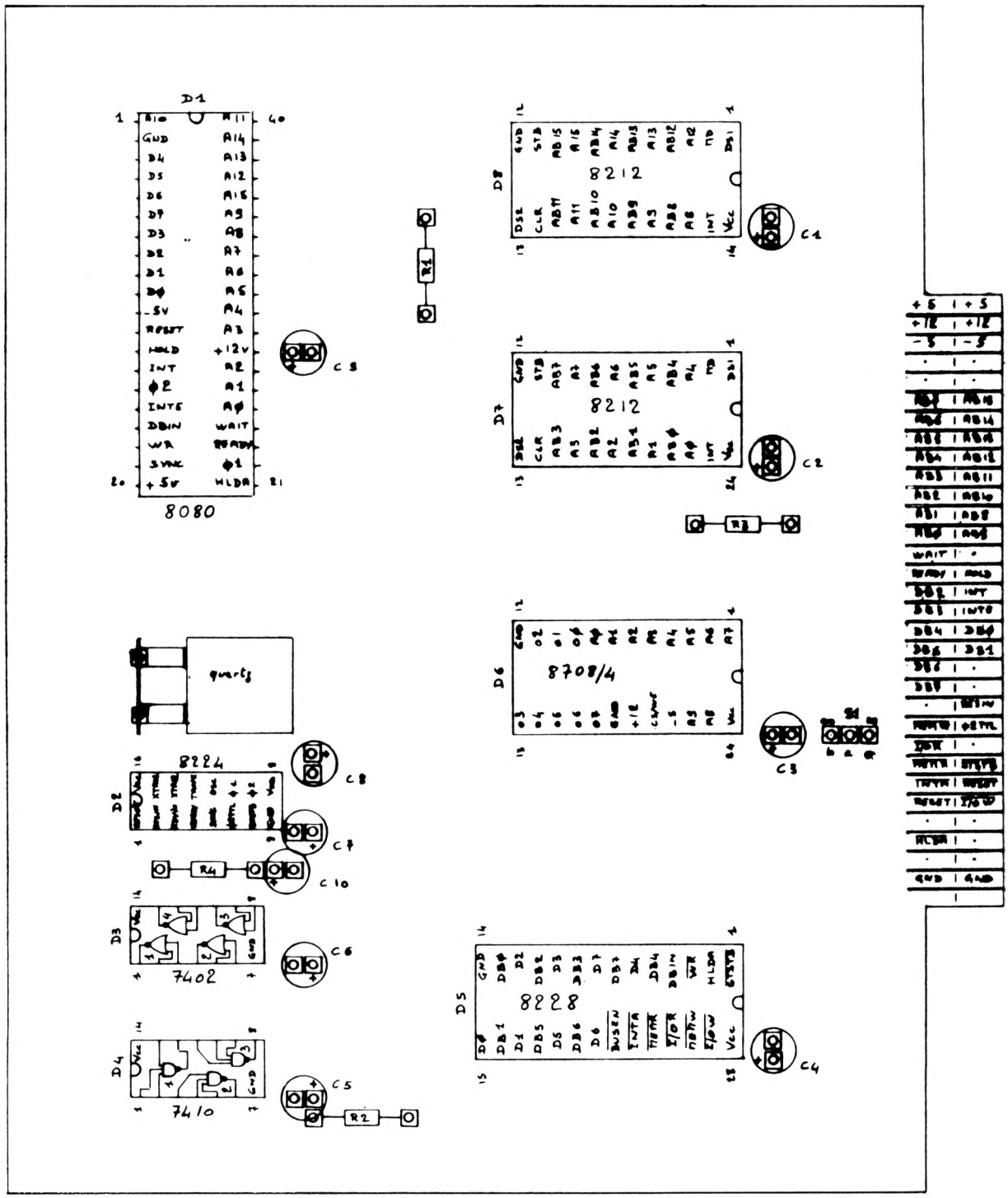
Carte 7000, côté composant, les 10 trous à gauche

Souder les trois condensateurs goutte tantale : la tache vers le haut pour les deux de droite, vers la gauche pour celui du centre. Les quatre diodes de droite ensuite, anneau blanc vers la droite. Les quatre condensateurs de 4 700 μF 16V ont tous le + vers le bas. Le gros de 4 700 μF 25V a le + à droite (le souder après les quatre diodes de gauche, qui ont le + à droite ; rabattre ces diodes à

C1 à C9 = 2,2 µf } tantale
C10 = 10 µf }

R1 R2 R3 1000 Ω
R4 3900 Ω

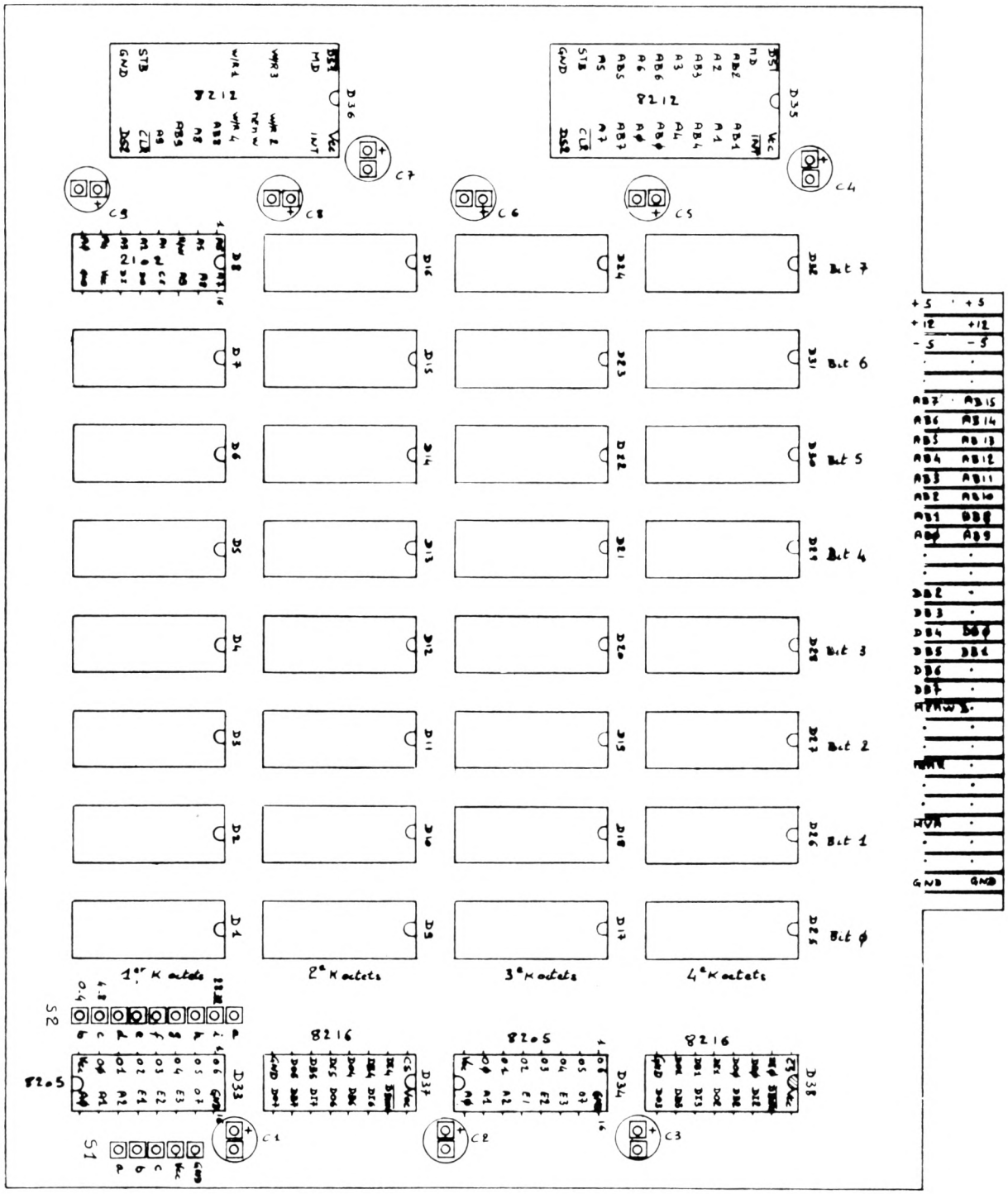
S1 a relié à b pour RPROM 8704 c non connecté
a relié à c pour RPROM 8708 b non connecté



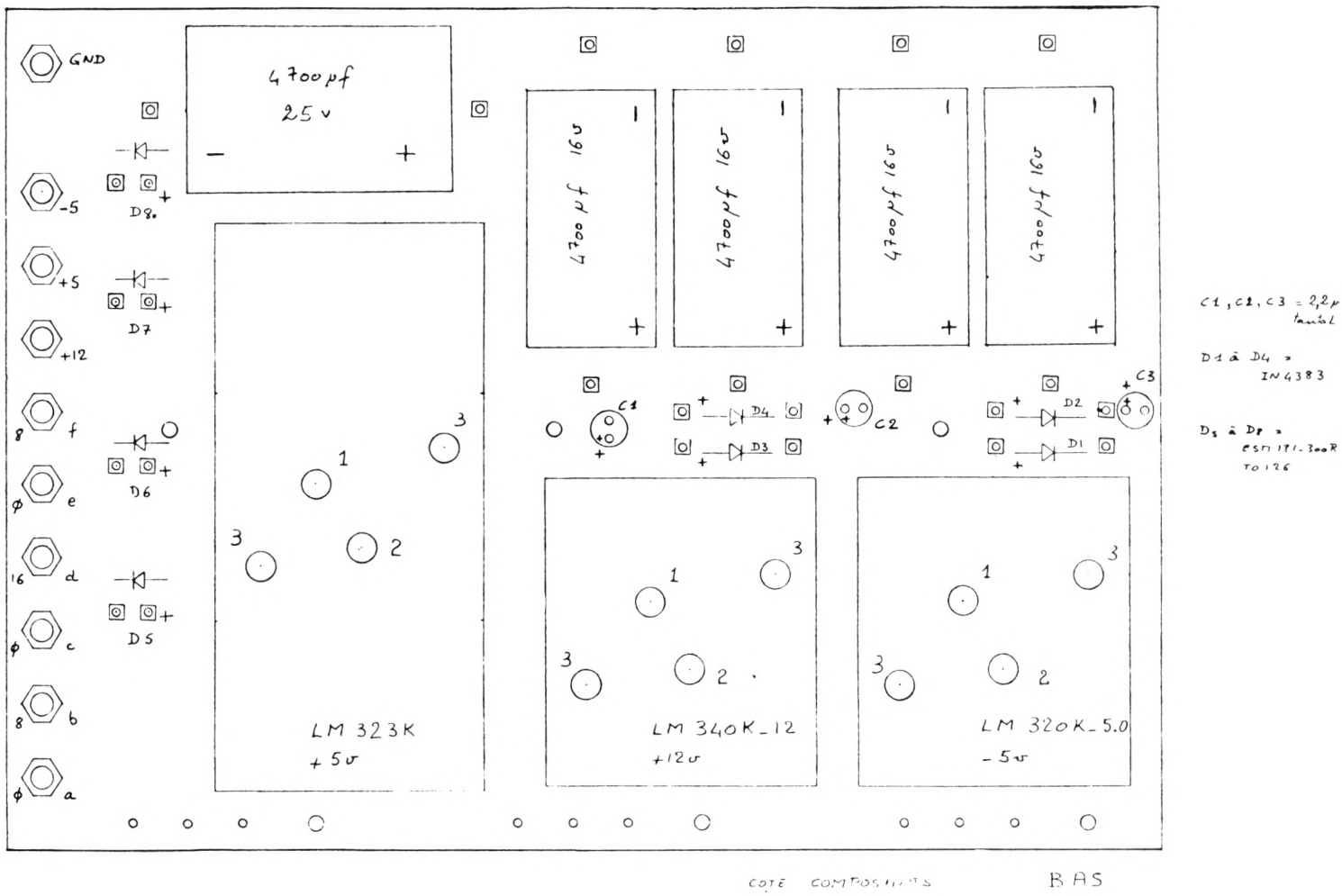
D1	8080	PROCESSEUR
D2	8224	HORLOGE
D3	7402	
D4	7410	
D5	8228	SYSTEME BUS
D6	8708/4	RPROM
D7	8212	BUFFER ADRESSE
D8	8212	

Carte processeur

D1 à D32	8108	D 37	8216
D 33	8205	D 38	8216
D 34	8205		
D 35	8212	C1 à C9	8,2 µf tantal
D 36	8212		



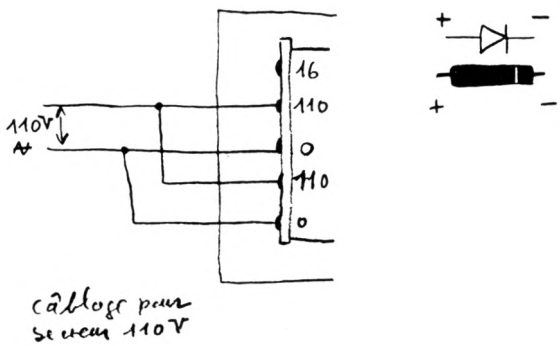
Carte mémoire 4 K statique

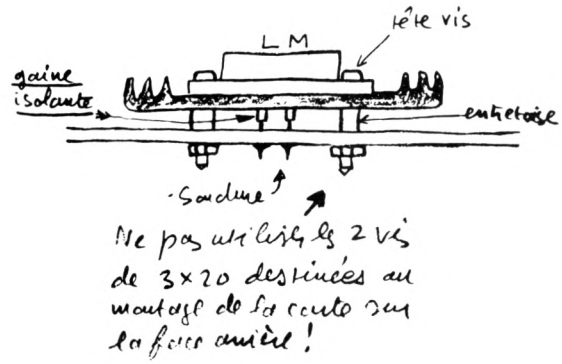


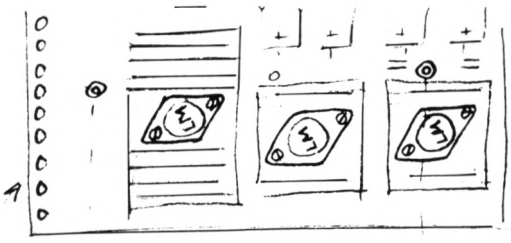
carte alimentation

plat vers le bas si vous avez le modèle rectangulaire à trou central, où le + est imprimé).

Attention ! → Enfiler sur les pattes des régulateurs LM des morceaux de gaine isolante de 5 ou 6 mm coupés sur du gros fil isolé, de façon à les isoler des radiateurs. Enfiler les pattes des régulateurs dans les trous centraux des radiateurs : trous et pattes étant excentrés, il n'y a qu'une position de montage possible.



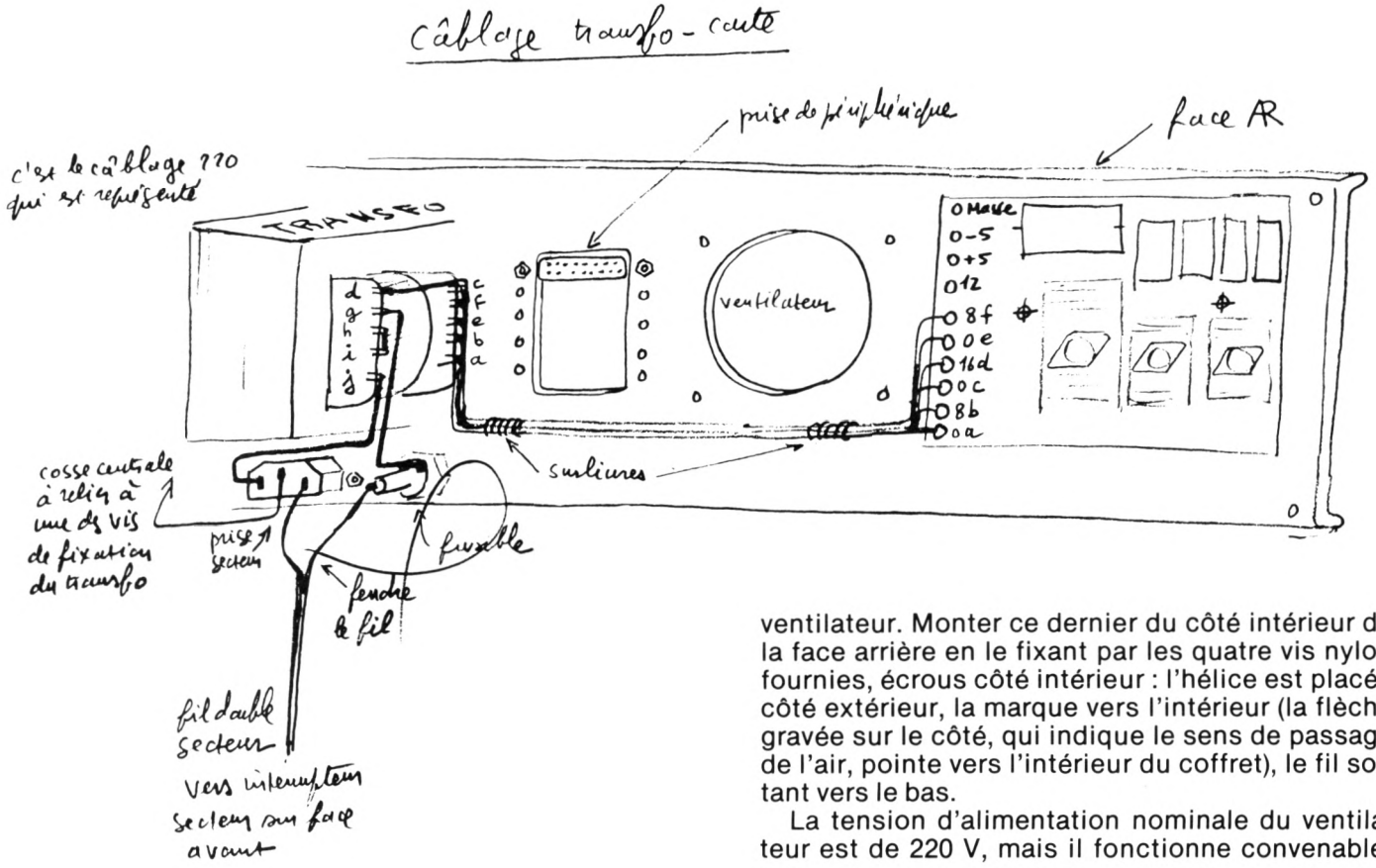




Relier les cosses du transformateur (soudure) aux bornes à vis de la carte (2e écrou pour serrer le fil, sans soudure) *lettre à lettre identique.*

h) *Ventilateur*

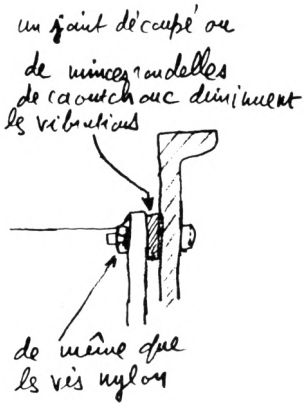
Si votre machine a une capacité mémoire supérieure ou égale à 4 K, elle doit être munie d'un



ventilateur. Monter ce dernier du côté intérieur de la face arrière en le fixant par les quatre vis nylon fournies, écrous côté intérieur : l'hélice est placée côté extérieur, la marque vers l'intérieur (la flèche gravée sur le côté, qui indique le sens de passage de l'air, pointe vers l'intérieur du coffret), le fil sortant vers le bas.

La tension d'alimentation nominale du ventilateur est de 220 V, mais il fonctionne convenable-

ment, à vitesse plus faible, sur 110V. Ses fils sont donc à relier normalement aux cosses g et j du transformateur. Si vous voulez réduire le bruit du ventilateur, si vous n'êtes pas sous les tropiques



et si votre machine ne dépasse pas 8 K, reliez les fils aux cosses i et j. La durée de vie du ventilateur sera encore augmentée.

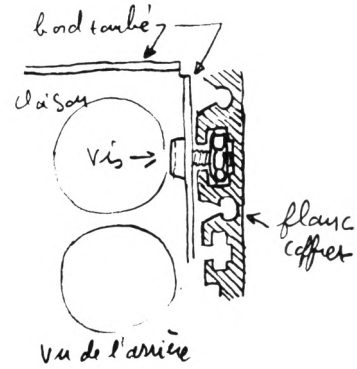
Si votre tension secteur est 110V, reliez obligatoirement les fils aux cosses i et j.

i) *Carte FOND DE PANIER : réf. 6000*

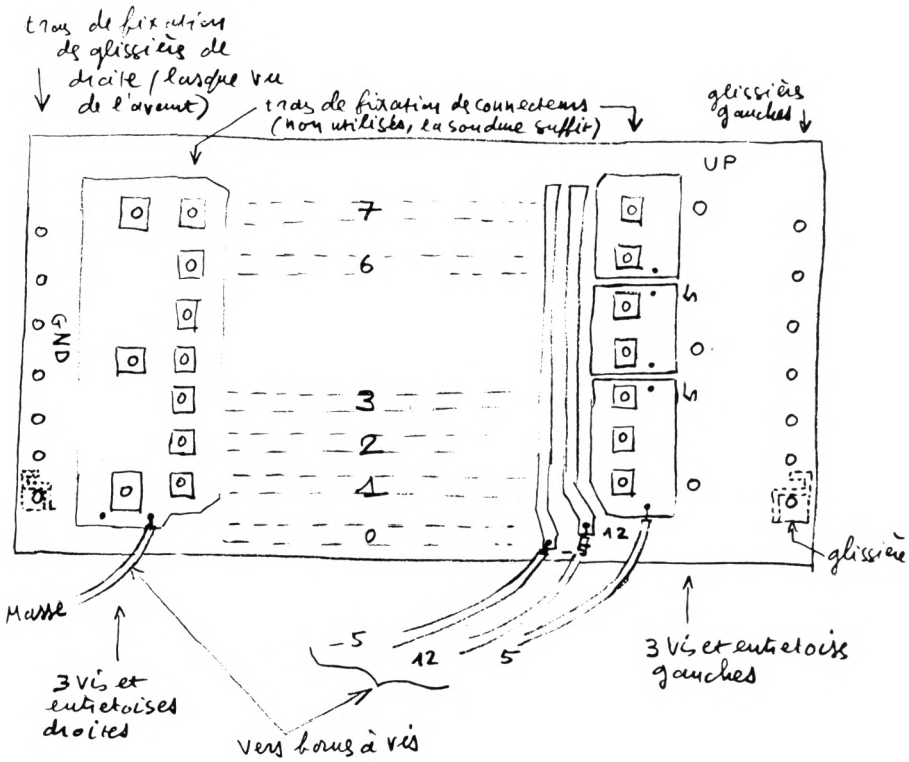
Le rôle de cette carte a été décrit en 6-b (voir le croquis). Le haut de la carte est repéré par UP (côté des inscriptions du bus). Huit doubles rangées de 2 x 31 trous chacune figurent sur la carte. La double rangée du bas sert à la liaison des deux fonds de panier pour les machines ayant plus de sept cartes (repère 0).

Placez votre premier connecteur dans la double rangée située immédiatement au-dessus (repère 1) : *le connecteur est à placer du côté non métallisé de la carte,* ses picots étant à souder côté pistes du bus. Attention aux excès de soudure, il en faut peu.

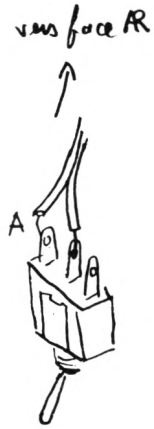
Placer ensuite le deuxième connecteur en 2, le souder, et ainsi de suite jusqu'au dernier.



Vous constaterez que la cloison touche presque le fond perforé du coffret, mais laisse plus d'un



centimètre de jeu au-dessus pour laisser passer l'air et les câbles de liaison des périphériques.



Fixer l'interrupteur à sa place sur la face avant, et serrez l'écrou en le positionnant de façon à ce que le levier débatte horizontalement et que la cosse A soit vers le flanc gauche du coffret. Le montage est terminé ! Aux essais.

j) *Mise sous tension*

Commencez par mettre sous tension un bref instant (n'oubliez pas de mettre le fusible), aucune carte n'étant insérée dans les connecteurs. Si le fusible ne saute pas, si rien ne fume, votre câblage est sans doute correct. Il peut arriver au premier allumage une petite bouffée de fumée, instantanément dissipée : un tel incident est dû à un fin pont de soudure dissimulé sous la couche de résine due au flux de la soudure, ce qui l'a rendu indécelable lors du contrôle visuel. Le pont de soudure, s'il se trouve en un point où il crée un court-circuit, se vaporise alors brusquement.

Si vous disposez d'un contrôleur universel de modèle courant, vérifiez les tensions aux bornes à vis supérieures de la carte alimentation : elles doivent correspondre aux valeurs désirées à quelques dixièmes de volts près (tolérances et erreur instrumentale). Attention à ne pas faire de court-circuit en prenant la mesure, appuyez-vous bien les mains sur la cloison centrale pour ne pas faire déraper une sonde entre deux bornes.

Vous pouvez aussi mesurer la consommation. Le plus pratique est de placer une prise multiple dans la prise murale secteur, puis d'y enficher une seule des broches de la prise mâle du cordon secteur d'Alcyane. Mettez l'interrupteur secteur du pupitre vers le voyant, sélectionnez la gamme alternatif 0,25 A (au plus) du contrôleur, plantez l'une des sondes dans le trou libre de la prise multiple et posez l'autre sur la 2e broche de la prise d'Alcyane. Ne vous électrocutez pas dans la manœuvre, elle est à vos risques et périls !

Sur notre propre machine (il y a des dispersions dues aux tolérances de consommation des composants, assez larges) nous avons mesuré, sur le secteur 220 V, une intensité de 50 mA à vide (ventilateur câblé en 110 V) et 173 mA en charge avec la carte processeur, une carte mémoire 4 K et le pupitre, ce qui fait... 38 watts !

Une chose est sûre, Alcyane ne vous ruinera pas en notes d'électricité : la moitié de la consommation de votre lampe de bureau...

Revenons aux procédures de mise sous tension : après le test à vide, coupez le courant, mettez une carte dans un connecteur et rétablissez le courant. Tâtez du doigt les DIP : ils ne doivent chauffer que modérément, le 8228 est un des plus chauds. Si un DIP vous brûle le doigt, coupez l'interrupteur secteur immédiatement (ne jamais retirer ou mettre une carte la machine étant sous tension), ce DIP est sûrement monté à l'envers.

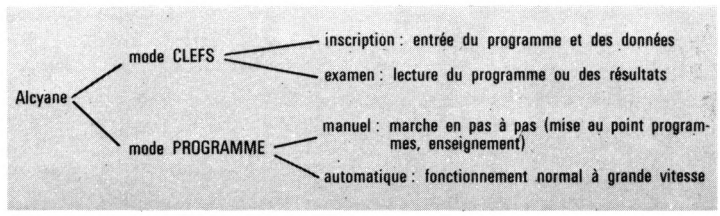
Maintenant tout est prêt, nous allons pouvoir commencer les manipulations au pupitre.

CHAPITRE 6
LA MANIPULATION DU PUPITRE

Alcyane a été conçue de façon à ce que l'unité centrale seule, sans aucun périphérique, puisse être utilisée de façon autonome. C'est la raison pour laquelle elle a été munie d'un grand nombre de commandes et de voyants lumineux sur sa face avant, qui constitue ainsi un panneau de contrôle ou «pupitre» très complet. Selon l'habitude, nous appellerons «clefs» les interrupteurs du pupitre (à l'exclusion de l'interrupteur «secteur» qui appartient à une catégorie différente). Une quantité importante de composants électroniques, située derrière la face avant, relie clefs et voyants au reste du système en matérialisant leurs fonctions.

6 - 1. LES MODES DE FONCTIONNEMENT

Deux modes de fonctionnement, ayant chacun deux variantes, existent sur Alcyane. Ils correspondent aux utilisations suivantes :



Dire «mode CLEFS» signifie que ce sont les manipulations des clefs qui gouvernent le fonctionnement de la machine : dans ce mode, le microprocesseur est isolé du reste du système, et l'on peut inscrire directement dans la mémoire les informations affichées à l'aide des clefs «information», l'adresse de l'emplacement où cette inscription se fera étant affichée à l'aide des clefs «adresse». Symétriquement, on peut examiner le contenu de la mémoire qui apparaîtra sur les voyants «information».

Dire «mode programme» signifie que ce sont les instructions du programme qui, exécutées l'une

après l'autre par le microprocesseur, gouvernent le fonctionnement de la machine. Leur exécution peut se faire à deux rythmes différents : en «manuel», le tythme sera déterminé par la succession des pressions du doigt sur la clef «pas à pas», en «automatique» il sera égal à la vitesse de fonctionnement normale d'Alcyane (environ 150 000 instructions par seconde).

Les voyants sont reliés au bus dont ils visualisent l'état : leur allumage ou leur extinction ne correspondra donc pas forcément aux positions des clefs.

La mise sous tension se fait par l'interrupteur secteur qui, contrairement aux clefs, se manœuvre horizontalement : cette disposition a été choisie pour éviter de couper accidentellement le courant par une fausse manipulation, ce qui bien sûr provoquerait l'effacement du programme entré en mémoire. Un capuchon plastique différencie encore l'interrupteur secteur, et le voyant «secteur» s'allume quand la machine est sous tension.

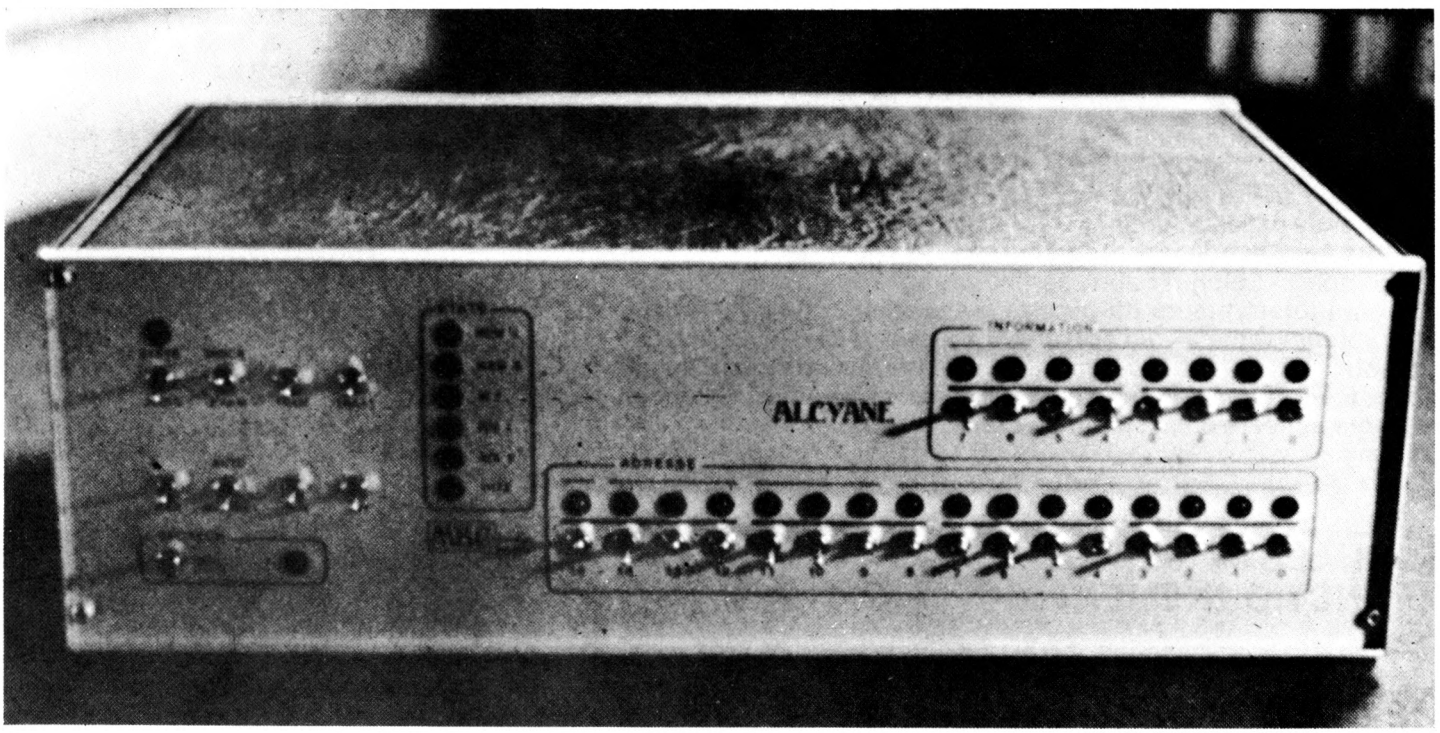
Contrairement à certains matériels, le fait de mettre en marche et d'arrêter Alcyane fréquemment n'entraîne aucune conséquence nuisible en ce qui concerne sa durée de vie — qui est indéfinie en pratique. Lorsqu'on vient de couper le courant, il ne faut pas le remettre immédiatement : laissez coupé une dizaine de secondes pour permettre aux condensateurs de filtrage de se décharger.

La carte processeur est munie d'une remise à zéro automatique du PC et des registres tampons de périphériques à la mise sous tension : on est ainsi assuré qu'un programme résidant à demeure dans une mémoire ROM commencera alors son exécution à l'adresse zéro, sans intervention manuelle. Une Alcyane à application spécialisée (par exemple pilotage d'un processus industriel) pourra ainsi être équipée d'une face avant ultra-simplifiée ne comportant en tout et pour tout qu'une serrure à clef plate pour la mise sous tension, et le voyant secteur.

Dans une version de base d'Alcyane sans mémoire ROM, si l'on met sous tension étant en mode programme manuel, on voit le RAZ s'effectuer en un éclair sur les voyants adresse. Si l'on est en programme auto, la machine démarre à grande vitesse à partir de zéro dès le RAZ effectué, et comme mémoire et registres ont un contenu à la mise sous tension, on exécute donc un programme aléatoire qui en général boucle sur lui-même ou stoppe sur un HLT (76) bloquant toutes les commandes, et il faut en sortir par une interruption. Le mieux est simplement de couper le courant, de se mettre soit en mode clefs soit en mode programme manuel, et de remettre sous tension quelques secondes plus tard. On est alors prêt à travailler au pupitre.

6 - 2. ENTRER, RELIRE ET EXECUTER UN PROGRAMME

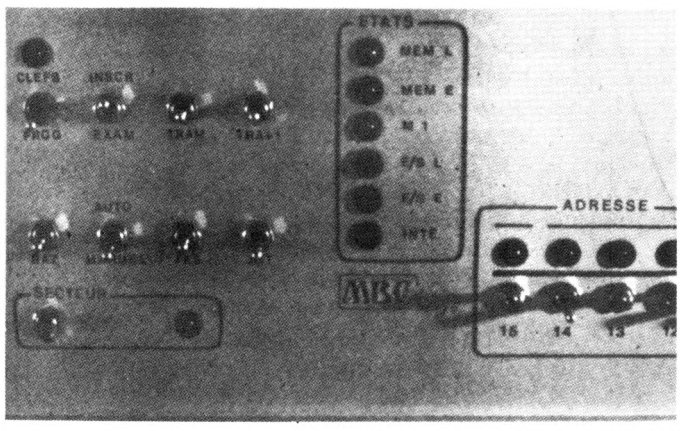
On demande le mode CLEFS en positionnant la clef en haut à gauche vers le haut, et on appuie une fois (ou deux si nécessaire) sur la clef PAS



pour faire allumer le voyant au-dessus du mot CLEFS. Lorsqu'il s'allume, c'est que l'on a pris le contrôle du bus et que l'on peut maintenant inscrire ou examiner. Pour entrer un programme ou des données (des informations, le mot couvrant les deux cas) on met la deuxième clef sur INSCR et l'on affiche aux clefs INFORMATION et ADRESSE respectivement l'octet à entrer et l'adresse mémoire où l'on veut le mettre : une clef basculée vers le bas indique un bit zéro, vers le haut un bit un. Comme vu par ailleurs, on travaille à partir d'un document rédigé en binaire, octal ou hexadécimal, et dans ces deux derniers cas on s'aide des marquages trois par trois ou quatre par quatre du pupitre.

L'affichage terminé et vérifié, on appuie sur TRAM (transfert à l'adresse manuelle). Fugitivement, le voyant MEM E (mémoire écriture) s'allume, et l'on revient en MEM L (mémoire lecture) ; le contenu de l'octet qui vient d'être inscrit en mémoire est relu et apparaît sur le bus données, c'est-à-dire les huit voyants INFORMATIONS, ce qui constitue un «contrôle par écho» de l'opération d'écriture. Si l'on appuie plusieurs fois sur TRAM, l'octet sera inscrit à nouveau au même endroit. L'adresse apparaît, elle, en bas.

En général on a à entrer plusieurs octets à des adresses consécutives (chargement d'un programme par exemple). Dans ce cas, le premier d'entre eux est entré comme nous venons de le voir, puis on affiche le suivant, on appuie sur TRA + 1, et ainsi de suite. TRA + 1 signifie «transfert à l'adresse plus un», fonction réalisée par les circuits intégrés de progression d'adresse qui l'incrémentent d'un unité puis exécutent le transfert.



Quand une séquence d'octets vient d'être entrée, il est prudent de la relire. Basculons la deuxième clef sur EXAM, puis appuyons sur TRAM : comme l'adresse initiale de notre séquence était restée affichée aux clefs, nous voyons apparaître le contenu du premier octet (et son adresse). En appuyant répétitivement sur TRA + 1, on contrôle le reste de la séquence. L'ensemble de ces manipulations se révèle très commode dès qu'on a acquis un peu de pratique.

A moins d'être très sûr de soi, il vaut mieux exécuter d'abord en pas à pas. On vérifie que la deuxième clef de la deuxième rangée est sur MANUEL, et on bascule la clef de sélection de mode sur PROG. Si l'on appuie alors sur PAS, on exécute un pas de programme, c'est-à-dire un macro-cycle, une instruction demandant un à trois pas pour s'exécuter. Mais où démarre-t-on ? A l'adresse qui était contenue dans le PC au moment où on est passé sur CLEFS. Ne parlons pour l'instant que du cas simple le plus courant, celui où l'on a chargé la séquence à exécuter à partir de l'adresse zéro : si le PC n'est pas à zéro, ce qu'on voit en regardant les voyants ADRESSE, on appuie sur RAZ et le tour est joué.

Ensuite, des pressions répétées sur PAS permettent de suivre le déroulement détaillé des instructions (voir chronique). Lorsqu'on se rend compte d'une erreur, on la corrige en repassant en mode CLEFS INSCR puis EXAM jusqu'à mise au point complète. On recopie alors le programme à la main en EXAM, ou on le liste en hexadécimal sur l'imprimante en l'écrivant en même temps le cas échéant sur mini-cassette ou sur disque.

L'exécution à vitesse normale a lieu sur PROG AUTO. Pour stopper le programme, on passe en MANUEL, pour le redémarrer on rebascule sur AUTO.

Commencez par le programme le plus simple. C'est un programme d'une seule instruction : JUMP à l'adresse zéro, l'instruction étant logée dans les trois premiers octets de la mémoire. Vous entrez successivement 11000011, puis deux fois des octets nuls, et vous exécutez en pas à pas et en auto : l'instruction JUMP demande trois macro-cycles, et l'on voit donc très bien la boucle. C'est le premier programme que nous avons chargé dans le prototype d'Alcyane à l'été 75 : grande a été l'émotion quand nous avons vu à l'oscillo le bus adresse repasser à zéro le troisième coup !

6 - 3. LA ROULETTE ELECTRONIQUE

Presque aussi simple que le précédent, et il a pourtant 32 instructions... Mettons-nous en CLEFS INSCR avec toutes les clefs du bus en bas, et appuyons sur TRAM puis sur TRA + 1 à répétition : on voit le comptage en binaire de l'adresse progresser régulièrement (une bonne méthode pour enseigner le binaire, d'ailleurs) et on arrête

quand 29 est allumé. On affiche alors le code du JUMP qu'on entre par TRA + 1, puis on affiche un octet zéro qu'on entre de même deux fois de suite : on est alors prêt à exécuter. Passons en PROG MANUEL, faisons RAZ et démarrons : nous exécutons une longue boucle de NOP terminée par un saut au début, les adresses évoluant de 0 à 32 comme on le voit bien sur le bus en AUTO, les six voyants de droite étant allumés ensemble par suite de la persistance rétinienne.

Si l'on passe en manuel on s'arrête quelque part... au hasard, car les NOP durant quatre cycles d'horloge on tourne à 500 000 instructions par seconde !

6 - 4. LA COMMANDE INT

Cet interrupteur à rappel permet, à partir du pupitre, d'interrompre le programme en cours. L'interruption ne peut bien sûr avoir lieu que lorsque le flip-flop INTE est armé par le programme, ce qu'indique alors l'allumage du voyant INTE. Une pression sur INT provoque l'exécution de l'instruction RESTART 7 (voir manuel instructions) ce qui permet de se brancher à un endroit prédéterminé dans le programme, avec sauvegarde du PC et sans affecter les contenus des registres tampons des interfaces de périphérique.

Ces derniers sont remis à zéro lorsqu'on utilise l'interrupteur RAZ en cours de déroulement de programme pour se brancher à l'adresse zéro.

6 - 5. VOYANTS D'ETAT

Il y en a six, et nous avons parlé de trois d'entre eux.

Les voyants Entrée/Sortie Lecture et Entrée/Sortie Ecriture s'allument lorsque les opérations correspondantes sont en cours d'exécution.

Le voyant M1 indique que l'on est dans un cycle M1 (lecture du premier octet d'une instruction) : il est inopérant dans les versions de base de la machine et demande une option spéciale pour être actif. En cours de pas à pas, l'examen des instructions qui défilent sur le bus informations permet de savoir facilement à quel moment on est dans un cycle M1.

Le lecteur avancé aura subodoré quand nous avons parlé de «microprocesseur isolé du reste du système» que la commande CLEFS envoie un signal HOLD vers le microprocesseur : en appuyant sur PAS on avance à travers un intervalle de sécurité, et quand le microprocesseur renvoie HLDA, le voyant CLEFS s'allume. Le bus est alors flottant, en état haute impédance, et l'on a réalisé un accès direct mémoire (DMA) manuel pendant lequel on pilote les lignes signaux et adresse du bus par les clefs.

La clef RAZ est prévue pour être utilisée seulement en mode programme. Si on appuie dessus par inadvertance pendant qu'on est en mode clefs, le microprocesseur reprend le contrôle du bus, et les commandes deviennent inopérantes. Il suffit alors d'appuyer sur la clef PAS, ce qui remet le microprocesseur... au pas. Bon travail au pupitre, et abandonnez-vous tout simplement à la fascination des yeux d'Alcyane, il est impossible de lui résister...

CHAPITRE 7
RETOUR AU HARDWARE

7 - 1. QUELQUES TESTS EN PAS A PAS

Nous avons décrit dans le chapitre précédent le montage du noyau de base d'Alcyane (l'unité centrale avec l'alimentation numéro un) sans trop chercher à décrire les procédures de tests (de fait, la majorité des kits montés tournent du premier coup à la mise sous tension). Mais pour ceux qui désirent en particulier aborder la maintenance premier échelon, avec un appareillage ne dépassant peut-être pas le voltmètre le plus courant, nous allons maintenant entrer un peu plus dans le détail.

En montant votre machine paisiblement avec soin (*festina lente...*) nos statistiques montrent que vous avez nettement plus de 95 % de chances de la voir tourner du premier coup. Si la distraction, ou les gremlins (verts, en général) s'en mêlent, il va vous falloir travailler un peu plus.

La première étape d'un test rationnel commence bien sûr, comme dant tout appareil électronique, par l'alimentation.

Ayant débranché les quatre fils supérieurs des bornes à vis, mesurons les tensions d'alimentation indiquées sur la planche de la carte 7000. Tout d'abord en alternatif sur les bornes du bas de a à f, pour vérifier que tout est correct côté transfo, fusible et interrupteur secteur, puis en continu sur les quatre bornes du haut. Si l'une d'elles est incorrecte, il y a une manœuvre à exécuter avant d'incriminer un composant.

Les soudures sur la carte 7000 sont en effet plus difficiles à faire que sur les cartes de logique pure, la capacité thermique de la métallisation et des pattes de composants étant supérieure : il est bon d'utiliser un fer beaucoup plus puissant, à panne plus grosse, et surtout de souder les pattes pour cette carte *des deux côtés de la carte.* Si donc l...alimentation paraît ne pas marcher correctement, faites ce complément de travail de soudure avant toute chose, et neuf fois sur dix tout rentrera dans l'ordre.

Sinon, un condensateur peut être défectueux : s'il est seul (sans autre capa en parallèle) et si l'une de ses connexions est coupée intérieure-

ment, la tension mesurée entre ses deux extrêmités (qui est la tension d'entrée du régulateur LM correspondant) sera trop faible, environ 7/10 de la normale, empêchant complètement le LM de fonctionner (il pourra donner zéro en sortie). Si la capa est en court-circuit larvé à l'intérieur (ou montée à l'envers par étourderie, toute prête à se transformer en bombe), la tension pourra chuter davantage : changer la capa qui chauffe (d'un modèle facile à trouver partout, et qui peut bien sûr être fournie aux isolés... géographiques).

Une diode défectueuse donne des symptômes voisins : en remplaçant la capa correspondante, on voit si un changement se produit ou non : si non, c'est une des diodes.

Les régulateurs linéaires de puissance LM sont des composants (de luxe) très rarement à incriminer. Véritables usines à gaz de précision, ils sont protégés à peu près contre tout : même un court-circuit franc à la sortie ne pourra pas les tuer, ils se couperont automatiquement. Si par exemple vous avez monté un des petits tantales à l'envers (il flambera peut-être) court-circuitant ainsi le LM, ce dernier n'aura en général subi aucun dommage : nous en verrons d'autres exemples plus loin. Il se peut également que le contact de masse par le boîtier d'un LM ne soit pas franc : une rondelle éventail sous une tête de vis, ou une cosse et un bout de fil arrangeront les choses.

Une fois tout en ordre avec l'alimentation à vide, branchez sur les bornes à vis les fils qui vont à la carte fond de panier (référence 6000), et remettez sous tension. Si l'une des tensions chute, c'est qu'il y a un court-circuit sur la carte fond de panier entre deux pistes (elles sont très serrées, attention au soudage des connecteurs) : grattez la soudure excédentaire après fusion si nécessaire.

Enfichez ensuite la carte liaison, et mettez à nouveau sous tension. Si l'une des tensions s'écroule, c'est qu'il y a un court-circuit sur l'une des cartes de la face avant. Dessoudez alors les fils d'alimentation de toutes les cartes sauf une, mesurez, rajoutez un autre si tout va bien, etc.

Dans une panne particulièrement bizarre, la machine chaque fois qu'on l'allumait fonctionnait bien pendant une dizaine de minutes, puis tous les LEDS de la face avant s'éteignaient progressivement. Une mesure montra qu'en même temps le 12 V s'effondrait. Quelle que soit la carte de face avant branchée sur la carte liaison, le résultat s'avérait identique, ce qui était pour le moins surprenant.

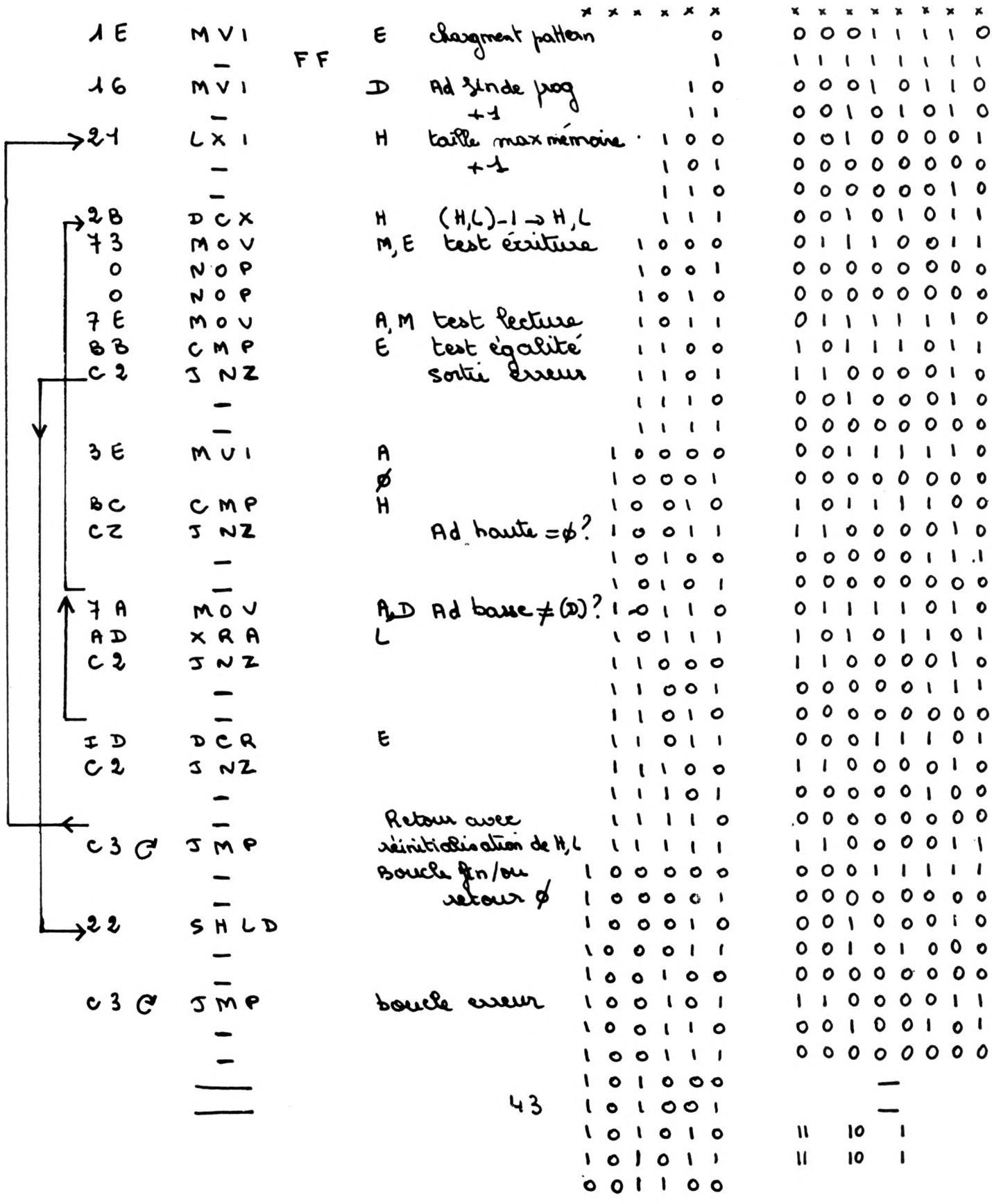
Une investigation plus poussée révéla qu'au montage du kit, les fils 12 V et 5 V venant de l'alimentation avaient été soudés sur la carte fond de panier sur le 5 V et le 12 V, dans cet ordre (effectivement, les chiffres métallisés sur la carte sont décalés par manque de place : il faut consulter le croquis du paragraphe i où les fils sont clairement marqués, en bas près de l'accolade, pour lever toute ambiguïté). Ainsi, toute la TTL et les LEDs de la face avant étaient alimentés en 12 V, et le malheureux LM 340 K, prévu pour débiter au maximum 500 mA, se mettait progressivement à genoux sous la charge... tout aurait dû flamber...

Eh bien, on peut dire que l'électronique d'Alcyane est solide : les deux fils furent intervertis sur les bornes à vis, et tout rentra dans l'ordre. Pas un seul circuit n'avait grillé, ni dans l'alimentation, ni dans la face avant ! Enfin, il vaut vraiment mieux ne pas abuser de ce genre d'amusettes... qui ne coûta fort heureusement que deux conversations téléphoniques entre Paris et la Charente. C'est du reste cet incident qui inspira à la victime le processus d'essai pas à pas par débranchement des câbles de la carte liaison que nous venons de décrire, qui est incontestablement utile dans les cas difficiles.

Outre les points, déjà signalés dans les paragraphes qui précèdent, concernant l'alimentation et le fond de panier, ainsi que la liste des neuf remarques du début, les problèmes qui peuvent se présenter sont en général dus :

— à un pontage de soudure entre deux pistes : il entraîne soit un court-circuit, soit un désordre logique,
— à un défaut de conduction (très rare) dans un trou métallisé (l'ohmmètre d'un contrôleur universel employé sur le calibre le plus faible sera utile pour vérifier le bon passage du courant : la résistance d'un côté à l'autre de la carte doit être absolument nulle ; attention si des composants sont déjà montés : utilisez alors la gamme de l'ohmmètre qui emploie une tension assez basse pour ne pas endommager certains circuits intégrés). Pour y remédier, passez un bout de fil dénudé très court à travers le trou, et soudez des deux côtés de la carte : la réparation est permanente et sans danger,
— à un fil cassé dans un câble plat au ras d'une carte donnant un contact intermittent, d'où un fonctionnement erratique : dénudez et ressoudez, en raboutant avec 2 ou 3 mm de fil nu si nécessaire. Une soudure sèche sur une broche de connecteur de la carte fond de panier produit des désordres analogues.

Test mémoire. Après avoir fait tourner les premiers petits programmes, en voici un — encore bien simple, vous pourrez le perfectionner — destiné à tester chaque circuit intégré mémoire avec un pattern changeant. En cas de discordance entre le contenu de l'octet écrit et le contenu de l'octet lu, il s'arrête après avoir stocké dans un coin l'adresse incriminée. Il n'y a plus qu'à le récupérer au pupitre en mode EXAM, puis à exécuter manuellement un court programme sortant A et E. A partir de là, un coup d'œil sur la planche de la carte mémoire 4K permet de localiser rapidement



Un programme de test mémoire

le point défaillant : la matrice des 32 DIPs est organisée en lignes et colonnes correspondant aux 1K, 2K, 3K, 4K et aux bits 0 à 7 des octets, comme indiqué. Le contenu de HL et le bit différent entre A et E indiquent le DIP à examiner. A neuf contre un encore il s'agira d'une soudure mal faite sur une des pattes, les mémoires 2102 étant d'une fiabilité et d'une robustesse remarquables.

Le test mémoire fourni sur disques est beaucoup plus complexe et dure plusieurs heures.

7 - 2. LA CARTE COMBINEE (REF. 4010)

Cette carte joue le rôle d'interface primaire (côté bus) pour les cinq périphériques ou interfaces finaux suivants :

— l'imprimante 20 colonnes électrorésistive,
— le clavier spécial en ordre alphabétique,
— la visualisation à plasma d'une ligne de 16 positions,
— premier interface magnétocassette,
— deuxième interface magnétocassette.

Comme sur d'autres cartes, vous trouverez progressivement de plus en plus de DIPs de la série 74 avec un code 74LS quelque chose. Cette nouvelle TTL, la Low power Schottky, a une consommation beaucoup plus faible et d'excellentes performances, mais est malheureusement plus chère que l'ancienne... D'autre part, tous les circuits 74 ne sont pas encore livrables en LS.

Le montage de la carte combinée est très simple, sans astuces particulières. Placez les 11 DIPs selon la planche indiquant leur implantation, et soudez-les au revers côté soudure. Faites de même avec huit condensateurs goutte au tantale (attention à la polarité) puis avec les deux résistances.

Restent maintenant deux cavaliers à positionner : le texte sur la planche indique de quelle façon ils permettent de donner à la carte son adresse (nous rappelons que dans Alcyane les adresses de cartes sont en décimal (BCD) de 00 à 19). Si vous devez utiliser le Basic, ou un software standard quelconque adressant cette carte, reliez sagement a et c pour S1, et a et o pour S2 : la carte aura ainsi l'adresse 00 standard.

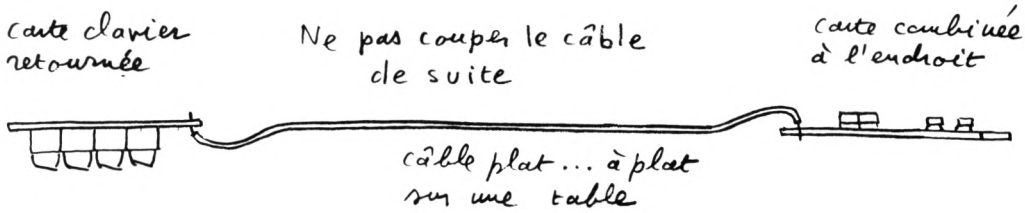
7 - 2 - 1. Liaison avec le clavier

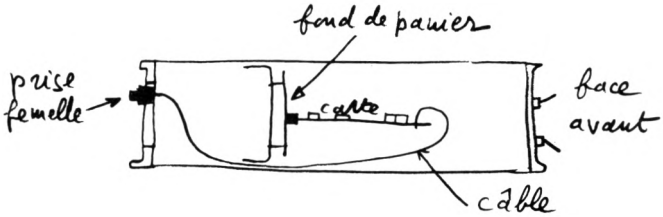
Il nous faut à présent relier la carte aux divers périphériques. Nous allons décrire ici le raccordement au clavier (imprimante et visu seront traitées dans leur chapitre).

Il se fait à l'aide d'un câble plat à 14 conducteurs longs de un mètre, venant du clavier, dont les fils doivent être soudés aux plots JLNFMKHED $CBAGV_{cc}$.

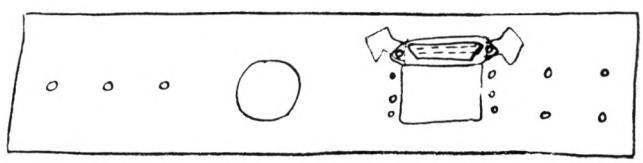
Le premier croquis indique la position du câble plat. On vérifiera sans peine la concordance des fils d'alimentation (avec Vcc, le + 5 V).

Le second croquis indique où faire passer le câble dans le coffret de l'unité centrale, par dessous la cloison transversale : présentez la carte au-dessus du coffret pour mesurer le câble, en laissant bien la longueur de la boucle, et coupez-le à l'aplomb de la face arrière.





Opérez de même pour les autres câbles, puis montez les interfaces cassettes s'il y en a (voir ci-après).



La prise à 36 contacts auto-nettoyants dorés doit être montée ensuite, horizontale, de telle

façon que le V formé par ses flancs latéraux pointe vers le bas.

7 - 2 - 2. Fonctionnement de la carte combinée et du bus

Cette carte, spécialement étudiée pour la connexion des périphériques les plus courants d'Alcyane, est intéressante à détailler en tant que fonctionnement et nous permettra de rafraîchir les connaissances en logique acquises en 2-2. Nous nous référerons tout du long à la grande planche indiquant son schéma logique, à l'aide des symboles que nous avons étudiés.

On sait bien que l'existence du bus permet de rendre pleinement modulaire une machine, dès lors entièrement constituée d'une série de cartes (les modules) enfichables directement dans les connecteurs reliés chacun à la totalité des lignes du bus. Pour étendre la machine, en capacité mémoire, en fonctions nouvelles ou en interfaces variés, on y ajoute a posteriori des cartes supplémentaires. Qui plus est, on peut même permuter leurs positions dans les connecteurs sans influer sur le fonctionnement logique (nous en verrons bientôt un des avantages). Comment une modularité aussi poussée est-elle possible ?

L'adressage

C'est une convention d'adressage généralisé qui est bien sûr la clef de cette flexibilité. Elle doit être établie de façon à permettre au microprocesseur :

— d'adresser individuellement chaque module (le *sélecter*),
— de lui envoyer des commandes,
— d'adresser éventuellement des emplacements précis dans le module (carte mémoire par exemple, tampon de l'interface TV, etc.),

tout ceci en compliquant le moins possible le matériel pour ne pas grever le prix de revient.

L'un des premiers problèmes qui se pose est de savoir comment différencier une adresse mémoire d'une adresse de périphérique de façon commode pour l'utilisation des signaux. En effet, deux types d'information peuvent se trouver sur les 16 lignes du bus adresse :

— *une adresse mémoire* de 16 bits, permettant d'adresser 64K octets. Une telle adresse apparaît lors de la lecture en mémoire d'une instruction à exécuter, ou d'une donnée, ou lors d'une écriture en mémoire.

— *une adresse périphérique*, contenue dans une instruction IN ou OUT (deuxième octet). Dans ce cas, cette adresse de 8 bits est répétée deux fois par le microprocesseur, en parallèle, sur le bus adresse. Ainsi, les bits A_8 à A_{15} prendront les mêmes valeurs que les bits A_0 à A_7 (sur les planches d'Alcyane, les lignes correspondant au bus adresse sont repérées de AB_0 à AB_{16}, pour Adress Bus).

Une convention intéressante avec le 8080 est d'utiliser le bit A_{15} pour distinguer ces deux cas :

— si $A_{15} = 0$, il s'agit d'une adresse mémoire
— si $A_{15} = 1$, il s'agit d'une adresse périphérique.

Ce faisant, on limite l'adressage mémoire de 0 à 32K (les études de marché des microprocesseurs montrent que 90 % environ des systèmes se situent dans cette fourchette), mais on simplifie le hardware : nous l'appellerons convention petit système.

Pour des raisons de simplicité, c'est cette solution qui est employée dans les versions courantes d'Alcyane (un autre type de carte processeur doit être utilisé pour aller au-delà de 32K).

Les considérations d'adressage qui suivent maintenant sont valables pour les deux systèmes, les différences étant précisées.

Commande des modules

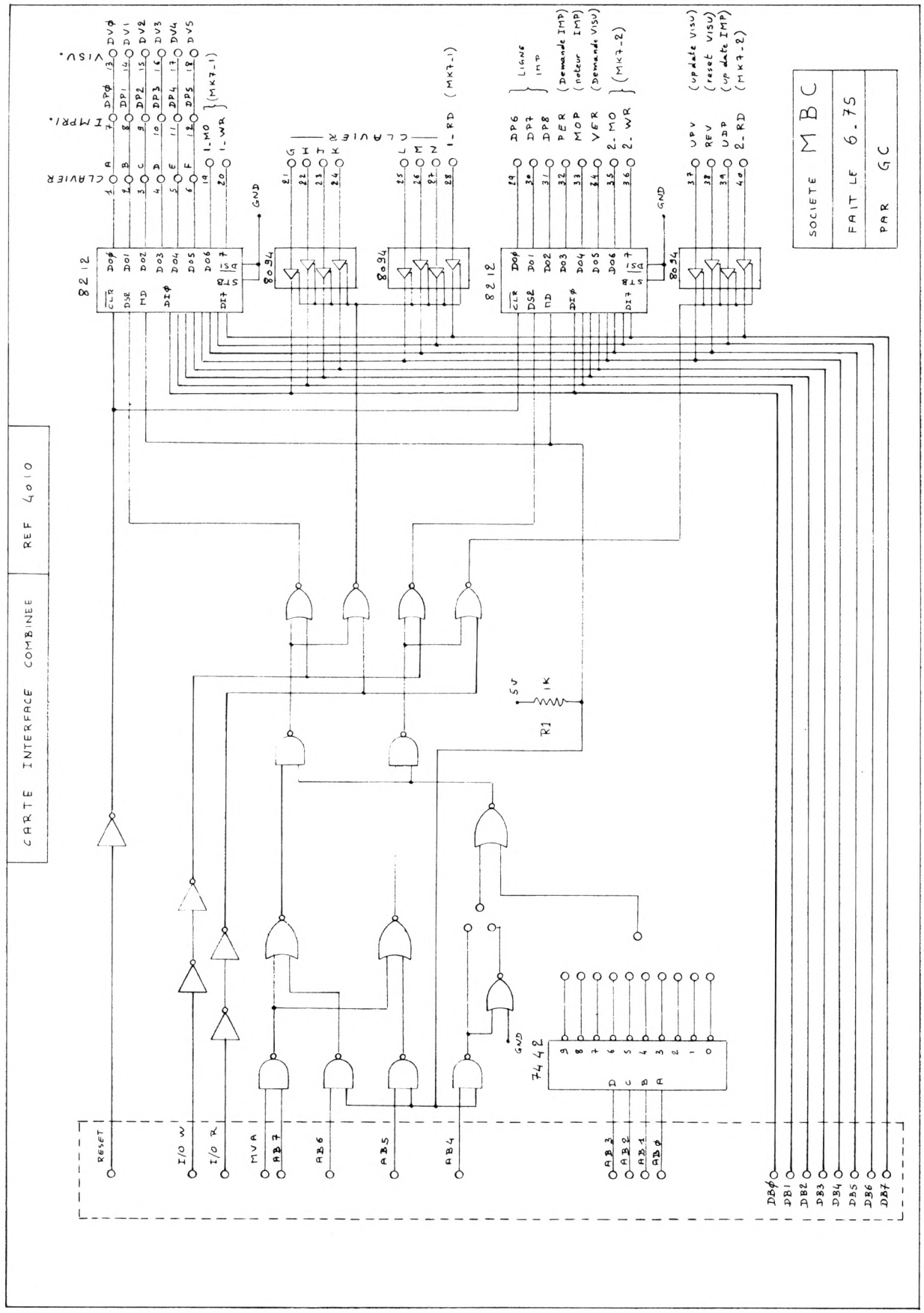
Avec la convention petit système, lorsqu'on adresse un périphérique, le bit A_7 sera systématiquement à 1, et l'on dispose donc des bits A_0 à A_6 qui seront utilisés de la façon suivante :

— A_0 à A_3 : ces quatre bits définissent en Décimal Codé Binaire le chiffre décimal des unités de l'adresse du module
— A_4 : ce bit indique le chiffre des dizaines de l'adresse. On peut donc adresser 20 modules, de 00 à 19.
— A_5 et A_6 : ces deux bits commandent les fonctions à réaliser au sein de chaque module, comme nous allons le voir plus en détail sur l'exemple de la carte combinée.

De la théorie à la pratique

Sur la gauche du schéma figurent les lignes reliées au bus : les 12 lignes supérieures sont unidirectionnelles et véhiculent des signaux toujours en provenance du microprocesseur. A ce sujet, notons bien que le bus adresse (dont nous retrouvons la moitié ici) présente à cet égard une différence fondamentale par rapport au bus *informations* (ou *données*, ou *data*...)

Ce dernier, dont nous voyons les 8 lignes dans le bas, est bi-directionnel : tantôt il envoie des signaux du microprocesseur vers la mémoire ou les périphériques, tantôt en sens inverse pour une lecture. Conceptuellement et électroniquement, ce n'est pas du tout la même chose...


CARTE INTERFACE COMBINEE
REF 4010
SOCIETE MBC
FAIT LE 6.75
PAR GC
CLAVIER
IMPRI.
VISU.
8212
8094
7442
5V
1K
R1
GND
RESET
I/O W
I/O R
MVA
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7

Les lignes AB_0 à AB_3 entrent dans un 7442, membre de la série 74, la grande famille logique TTL (prononcez «soixante quatorze quarante deux» et non sept mille quatre cent quarante deux, ce qui risquerait de dévoiler d'un coup à vos interlocuteurs les profondeurs d'une honteuse inculture électronique...). Ce circuit intégré est un décodeur décimal : à la réception d'un chiffre BCD il émet un zéro sur celle des dix lignes de sortie qui lui correspond.

Remarquons à ce sujet qu'en série 74 la valeur logique 1 (+ 5V) correspond sur une ligne à un débit quasi nul, tandis que le zéro logique (zéro volt) correspond à un passage de courant, circulant *de l'aval à l'amont* : le circuit aval qui reçoit le zéro voit ses transistors débloqués et émet un courant normalisé de 1,6 mA qui est la fameuse *charge TTL.* Ce courant est absorbé par le circuit amont qui est très précisément capable d'absorber un total de 16 milliampères, soit dix charges TTL. Cette norme de la famille permet donc une sortance égale à dix (un circuit amont peut être relié directement à un maximum de dix circuits aval travaillant en parallèle). Enfin, selon les cas les boîtiers sont inverseurs ou non inverseurs, c'est-à-dire qu'ils inversent ou non la valeur logique du signal d'entrée.

Revenons à D_2 : le cavalier reliant l'une des dix pattes de sortie (numérotées non consécutivement à cause de la présence de pattes d'alimentation) au plot situé en face définit le chiffre des unités de l'adresse de la carte. En standard, ce sera zéro.

La ligne AB_4 indique le chiffre des dizaines, zéro ou un. Elle entre dans une porte NON ET (l'une des quatre du DIP D_1 qui est un 7400) conjointement avec une ligne reliée par une résistance de mille ohms au + 5V, et par conséquent en permanence à 1. Comme une porte NON ET donne toujours 1 en sortie sauf lorsque les deux entrées sont à 1 en même temps, lorsque AB_4 vaudra 0, nous aurons 1 en sortie : la porte ainsi connectée joue le rôle d'un inverseur, NON.

En aval nous trouvons une porte NON OU et un cavalier. La porte NON OU donne toujours 0 en sortie, sauf lorsque les deux entrées sont égales à 0 en même temps. Or l'une des deux entrées, reliée à la masse (GND, ground), est à 0 en permanence. Si donc l'autre est à 1, on a 0 en sortie, si elle est à 0, on a 1 en sortie : le comportement est encore celui d'un inverseur, comme précédemment, mais par des moyens différents.

Si D_2 reçoit la bonne adresse, son cavalier va appliquer un zéro sur une des entrées de la porte NON OU qui le suit. Si l'adresse des dizaines est bonne (par exemple 0 et que le cavalier est en bas), la deuxième entrée de ladite porte NON OU sera *aussi* à zéro, et la sortie donnera 1 : ce 1 «bonne adresse» est appliqué à l'entrée de deux portes NON ET qui reçoivent par ailleurs des signaux provenant d'un peu plus haut. Voyons ce qui se passe de ce côté.

AB_5 et AB_6 sont inversés dès la première couche de portes, puis entrent chacun dans une porte NON OU recevant ensemble NON ($\overline{MVA}$ et AB_7). Sur petit système, $\overline{MVA}$ n'existe pas et est donc normalement à 1 (la porte qui suit ne pouvant faire circuler de courant sur une ligne en l'air). Supposons maintenant que pour la suite AB_7 vaille 1 (c'est donc qu'on a une adresse de périphérique), la porte donnera en sortie pour NON (MVA ET AB_7) un 0 que recevront les deux portes NON OU en même temps que $\overline{AB_5}$ et $\overline{AB_6}$ qui mariés au 1 «bonne adresse» traverseront les portes NON ET et iront sur D_5.

De plus haut viennent enfin les signaux $\overline{I/O\ W}$ et $\overline{I/O\ R}$ (NON écriture E/S, NON lecture E/S) qui après deux inversions, c'est-à-dire sans changement, attaquent eux aussi, mais deux par deux, les quatre portes NON OU superposées D_5. Supposons que $\overline{I/O\ W} = 0$, c'est-à-dire que le microprocesseur signale une écriture sur périphérique (instruction OUT), et que $AB_6 = 1$ et $AB_5 = 0$: dans ce cas, comme D_5 reçoit $\overline{AB_6}$ sur les deux portes du haut, elles auront déjà chacune un zéro en entrée. Celle du haut recevra en outre un deuxième zéro provoqué par la commande d'écriture : le 1 en sortie *sélectera* le 8212 du haut. Le 8212 est une bascule à verrouillage qui va alors happer le contenu du bus data, puis le présenter de façon stable sur ses huit lignes de sortie, simultanément vers le clavier, l'imprimante, la visu et l'interface cassette 1 (le 8212 est unidirectionnel). Ce contenu du bus data n'est autre que celui de l'accumulateur pendant l'exécution de l'instruction OUT.

Les deux 8094 qui sont en dessous ne seront pas actifs, puisque $\overline{I/O\ R} = 1$ et que la deuxième porte de D_5 aura donc un 1 en entrée, donnant 0 sur les commandes des 8094. Quant aux deux DIPs suivants, le fait que $AB_5 = 0$ les inactive de la même façon.

Revenons à notre 8212 du haut : sur six lignes il peut :

— envoyer un caractère ASCII 64 sur l'imprimante
— idem sur la visu ligne
— envoyer une configuration telle que 110111 sur le clavier, comme déjà vu, avec le zéro (appel de courant) baladeur

et sur les deux lignes suivantes

— envoyer une commande de mise en route moteur magnétocassette 1 à travers son interface
— envoyer un bit à MK7-1 idem.

Lorsqu'un caractère est envoyé sur l'imprimante, visu et clavier le reçoivent aussi. La visu ne le prendra pas en compte si un bit 1 préalable n'a

pas été mis sur la ligne 34, quant au clavier il est inerte et on n'a pas à s'en occuper.

Si l'on balaye le clavier, après avoir mis le motif dans le 8212 (il y reste), ou va lire par un IN ce qui en revient. Les 8094 sont des quadruples portes trois états, unidirectionnels, tous placés ici dans le sens périphériques vers microprocesseur. Ils ont tous leurs sorties reliées au bus données : comment ceci peut-il donc marcher ? Une telle connexion n'est possible que grâce à la logique *trois états*. Outre le 1 et le 0, une porte 3 états peut être mise en état haute impédance par une ligne de commande : elle devient alors complètement isolante, et il suffit lorsque n portes débitent sur un même point de les mettre toutes en haute impédance, sauf celle à qui l'on veut faire envoyer l'information. Une telle disposition (voir croquis) est impossible avec la TTL classique où les transistors des portes débiteraient les uns dans les autres sans frein... et rendraient l'âme rôtis vifs. La logique 3 états est donc précieuse pour simplifier les montages où de nombreux DIPs sont branchés sur les mêmes lignes. Les 8094 ne sont pas des bascules à verrouillage et l'information ne fait que les traverser.

Pour le clavier donc, après avoir chargé le 8212 par un

OUT 11000000

en voyant le pattern stocké dans ACC, on fera un

IN 11000000

et c'est la présence cette fois du $\overline{I/O\ R} = 0$ (et donc du $\overline{I/O\ W} = 1$ en même temps) qui évitera de sélecter le 8212 de nouveau, et au contraire débloquera simultanément les deux 8094 précédemment en haute impédance : les lignes 21 à 28 seront alors temporairement branchées sur le bus, et après inversion du passage dans les 8094 les huit bits correspondants arriveront dans l'accumulateur du 8080. Les sept bits de droite viendront du clavier, le bit le plus à gauche venant du magnétocassette 1 : on extrait par masquage ou autre ceux que l'on désire.

De façon analogue on pourra lire ou écrire sur les périphériques des bits isolés, ou des caractères, ou envoyer des commandes plus complexes, comme pour l'imprimante à 20 colonnes par exemple.

Cette imprimante électrorésistive brûle par étincelage une fine pellicule d'aluminium couchée sur une bande de papier noir. Chaque caractère est représenté par une mosaïque de sept lignes de cinq points dont le dessin est stocké dans une ROM. Ce sont les bits DP6, DP7 et DP8 qui indiqueront à cette ROM les numéros successifs des sept lignes de points dont la configuration sera envoyée sur les transistors envoyant les impulsions sur les 100 électrodes d'étincelage.

La carte combinée est en fait... une combinaison de hardware et de software particulière, les drivers des périphériques variés qu'on lui connecte.

Une remarque de nature électronique pour finir : l'utilisation de *Low power Schottky* est prévue sur cette carte pour diminuer les consommations. En effet, une charge standard en LS n'est que de 0,2 mA au lieu des 1,6 mA de la TTL normale, ce qui est un avantage important. Par contre la sortance est plus faible, et implique des contraintes au niveau conception logique visibles sur la carte.

7 - 3. LE CLAVIER SPECIAL

La présence du microprocesseur a permis de réaliser un clavier léger de très faible encombrement, pratique pour travailler d'une seule main. Il se présente sous la forme d'un ensemble compact de 40 touches, disposées en cinq rangées de huit, qui est vu par le microprocesseur comme une matrice de contacts à cinq lignes et huit colonnes : le *driver* du clavier (le software de gestion du clavier, si l'on préfère !) envoie à cadence rapide, sur la carte interface combiné, des octets dont la configuration est :

0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	0

et on recommence...

Le zéro balaye ainsi les huit colonnes l'une après l'autre. Après chaque envoi d'octet, on lit sur la carte combinée les cinq bits correspondant aux cinq rangées : si l'un d'eux est zéro, c'est qu'une des touches de la rangée correspondante est enfoncée, la colonne étant déterminée par la place du zéro dans l'octet. Le driver du clavier consulte alors sa table de conversion et restitue le code ASCII correspondant.

Si l'on a préalablement appuyé sur la touche *préfixe* (la deuxième de la rangée du bas), le driver ira chercher, lors de la pression de touche suivante, le code dans une seconde table décalée en mémoire de 40 positions par rapport à la première : cette seconde table contient les codes ASCII correspondant aux symboles placés à la partie supérieure des touches portant deux symboles.

Tel que gravé, le clavier comporte les 64 symboles USASCII normalisés classiques. Mais si on

Programme de prise en compte clavier

```
3000  LXI  B,0000
3003  PUSH B
3004  CALL 3025     } synchronisation clavier
3007  ØRA  A        }
3008  JNZ  3004
300B  CALL 3025     }
300E  ØRA  A        } scrutation du clavier
300F  JZ   300B     }
3012  PØP  B
3013  CPI  2A       si 2A dans l'accumulateur  "shift", redéfinition du clavier
3015  JNZ  301D
3018  MVI  C,2F
301A  JMP  3003
301D  ADD  C
301E  MØV  C,A
301F  LXI  H,30F7   } calcul de l'adresse du caractère ASCII dans la table
3022  DAD  B        } correspondant à la valeur de A
3023  MØV  A,M
3024  RET
```

Sous-programme scrutation clavier

```
3025  MVI  B,06
3027  MVI  D,DF
3029  MØV  A,D
302A  ØUT  C0       } 1 bit "zéro" sur la porte de sortie par permutation circulaire
302C  MVI  C,07
302E  IN   C0       }
3030  RLC           }
3031  RLC           } recherche sur la porte d'entrée d'un bit "zéro" activé par le
3032  JNC  3042     } contact d'une touche du clavier
3035  DCR  C        }
3036  JNZ  3031
3039  MØV  A,D
303A  RRC
303B  MØV  D,A
303C  DCR  B
303D  JNZ  302A     }
3040  XRA  A        } pas de touche enfoncée      retour avec "0" dans A
3041  RET
3042  MØV  A,B      }
3043  RLC           }
3044  RLC           } calcul de la position de la touche - dans l'accumulateur A
3045  RLC           } résultat de la matrice de câblage du clavier
3046  ØRA  C        }
3047  RET
```

>

Table ASCII - les caractères sont en ASCII codés hexadécimal

```
3100 41 44 47 4A 4D 50 53 20 42 45 22 2C 4E 51 54 20
3110 43 46 49 4C 4F 52 55 20 3D 2E 30 32 34 36 38 20
3120 20 20 31 33 35 37 39 20 20 20 58 0D 59 57 5A 4D
3130 4E 27 20 20 20 20 20 20 20 48 4B 27 20 20 20 20
3140 3B 3A 40 23 26 25 20 28 29 2B 2D 2A 2F 5E 20 5F
3150 20 24 5B 5D 3C 3E 20 20 20 5C 56 3F 20 21
```

>

Caractères de la table ASCII

```
ADGJMPS BE",NQT CFILORU =.02468   13579   X
¥WZMN'       HK'    ;:@#&% ()+-*/↑ ← $[]<>   \V? !
>
```

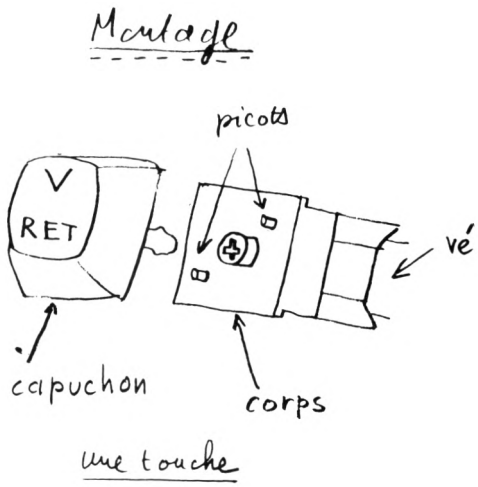
le désire, on peut affecter par software (en modifiant la seconde table du driver) le code qu'on désire aux 13 touches ne portant qu'un seul symbole : par pression préalable sur la touche préfixe, on pourra ainsi pour des applications spéciales disposer de 13 caractères particuliers, ou commandes supplémentaires.

Le driver comprend en outre des boucles d'attente et interblocages adéquats pour les rebondissements de contacts, etc.

Montage

Il consiste essentiellement à souder les touches sur la carte 8000 : le tout est de les positionner convenablement !

La carte 8000 est une carte simple face, comme la carte fond de panier, c'est-à-dire qu'elle ne porte de pistes métallisées que d'un seul côté, le côté *soudure*. Les composants (touches, résistances, câble plat) doivent être placés de l'autre côté sur la face vierge du verre-résine.



1. *Résistances*

Il y en a sept, à façonner de manière identique en coudant leurs pattes comme déjà vu avec un espacement de 12,7 mm.

Placez d'abord R7, côté nu, les pattes sortant côté soudure dans les deux pastilles carrées, puis R6, etc. jusqu'à R1. Toutes sont parallèles au grand bord de la carte. Soudez, coupez les pattes à ras.

2. *Touches*

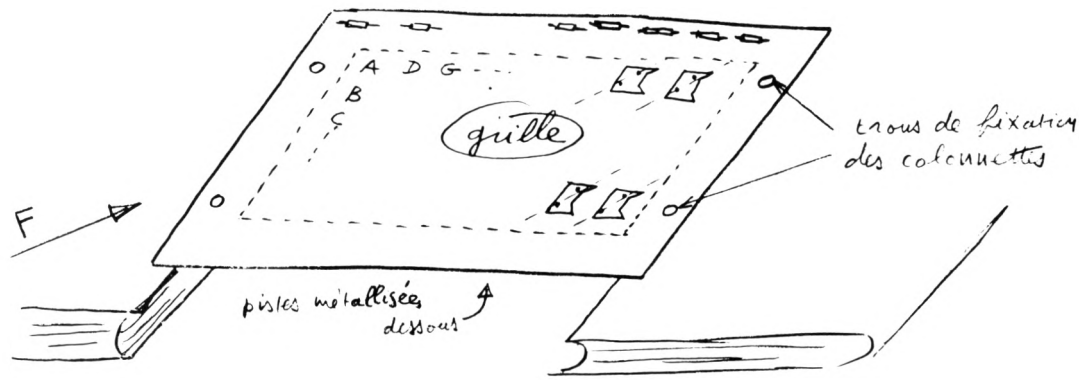
Placez la carte à plat sur votre table comme indiqué sur le croquis, les sept résistances sur le dessus, du côté le plus éloigné de vous ; et surélevez-la légèrement avec deux livres par exemple : elle vous apparaît alors orientée comme le clavier en utilisation. Les corps de touches sont à monter sur la grille de 160 trous (4 x 40 = 160).

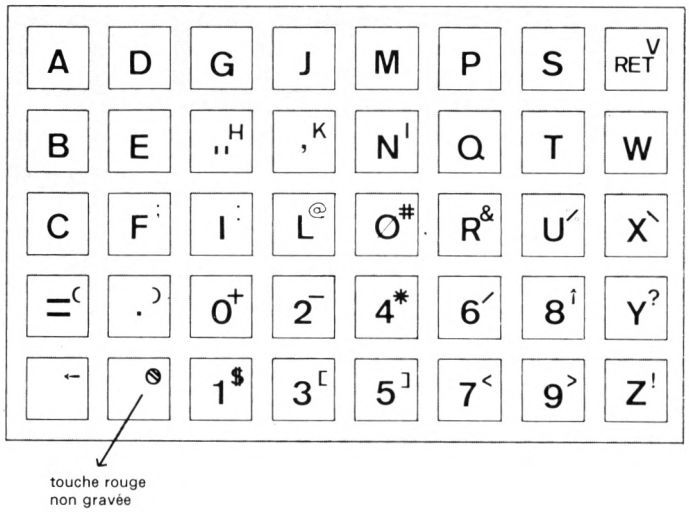
Commencez par la colonne de gauche. Chaque corps de touche doit être orienté de façon à ce que la face portant une échancrure en vé soit tournée *vers la droite*. Entrez trois des quatre pattes dans les trous, et poussez la quatrième patte avec une allumette pour la mettre en face de son trou : une fois entrée, la touche tient assez bien. Ne la soudez pas de suite, mais enfoncez-la bien à fond.

Colonne par colonne, mettez toutes les touches, puis vérifiez dans tous les sens qu'elles sont bien alignées. L'espacement entre les colonnes 4 et 5 est un peu plus grand que les autres, c'est normal.

Ne vous fiez pas aux chiffres imprimés sur les flancs des corps de touches pour l'orientation : ils ne sont pas toujours sur la même face. Par contre, les deux picots sur le dessus sont toujours dans le même sens : mirez la carte dans le sens de la flèche F, c'est la meilleure vérification.

Retournez alors la carte en douceur, posez-la à plat et soudez les 160 pattes. Vous remarquerez en soudant que deux seulement des pattes ser-



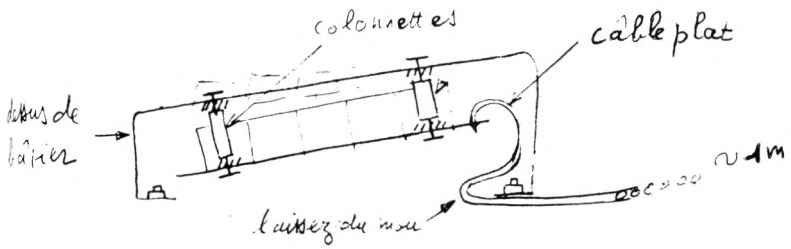


vent pour le contact, mais il est nécessaire de les souder les quatre pour la résistance mécanique.

Après cette opération, retournez la carte et enfoncez les capuchons des touches en vous guidant sur le croquis ci-dessous : on doit entendre «clic» à la mise en place.

3. *Câble plat*

Séparez une nappe de 14 conducteurs, dénudez à la pince et étamez les extrémités, ouvrez en éventail et enfoncez du même côté que les touches, comme déjà précisé dans les instructions générales.



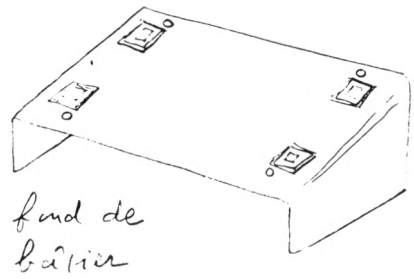
L'un des groupes de sept pastilles repéré FC2 se trouve entre R2 et R3. L'autre, FC1, au-dessus de R2, est composé d'un groupe de quatre et d'un groupe de trois dans le même alignement.

Soudez les 14 fils sous la carte, et repliez ensuite le câble comme indiqué sur le croquis. Ne pas couper le câble pour l'instant, laissez sa longueur intacte.

4. *Boîtier*

Au moyen des quatre colonnettes fixez la carte au-dessus du boîtier du clavier, à l'aide des huit vis de 3 x 10 et des huit rondelles papillon, à placer comme indiqué.

Retournez ensuite le fond en U du boîtier, et collez-y les quatre pieds dans la position indiquée sur le croquis.



Posez la partie déjà assemblée du clavier à plat, les touches par dessous, et coiffez-la du fond de boîtier.

Engagez les quatre vis de 4 x 18, et avant de les serrer vérifiez que le câble plat fait une légère boucle à l'intérieur du boîtier sans tirer directement sur les soudures : dès le serrage des vis, il sera solidement pincé entre fond et dessus de boîtier.

7 - 4. ALIMENTATIONS

Elles sont composées de modules + 5 volts et d'un module — 12 volts. Les modules 5 V servent à augmenter la puissance générale d'alimentation au fur et à mesure des extensions de taille mémoire, de l'addition de nouveaux interfaces, etc. Le module — 12 V sert à l'alimentation de composants spéciaux, comme par exemple les ROM contenant le graphisme pointillé des caractères pour la petite imprimante, l'interface TV et le clavier Azerty.

7 - 4 - 1. Module 5 V

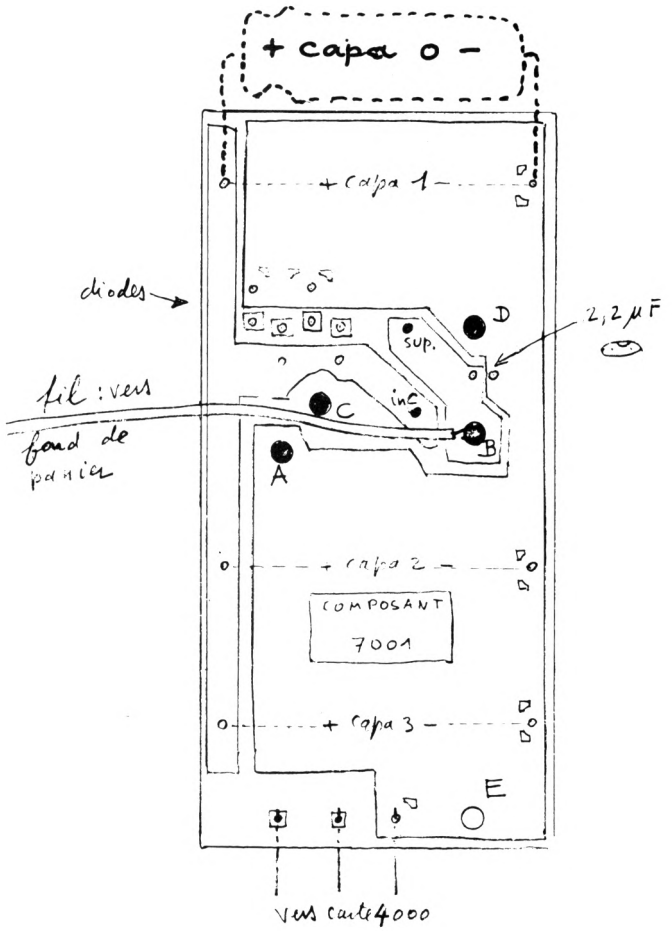
Placez devant vous la carte à plat, dans la position du croquis, pour repérer les jalons : vous lisez alors «composant, 7001».

Outre 21 petits trous, la carte est percée de cinq gros trous : un en bas, E, et quatre un peu au-dessus du milieu (A, B, C, D).

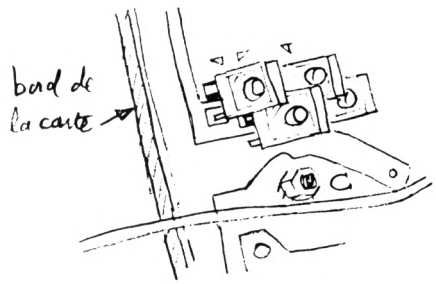
Voici maintenant le pas à pas de montage : il est facile et rapide, mais si vous ne respectez pas l'ordre, vous vous créerez des problèmes !

— Coupez 50 cm de fil rouge souple, dénudez un bout, étamez, passez dans le trou B et soudez *des deux côtés.*

— Placez les quatre diodes dans les quatre paires de trous en quinconce situés au-dessus de C : les faces métallisées des diodes doivent toutes être tournées vers la gauche (pattes fines vers C).



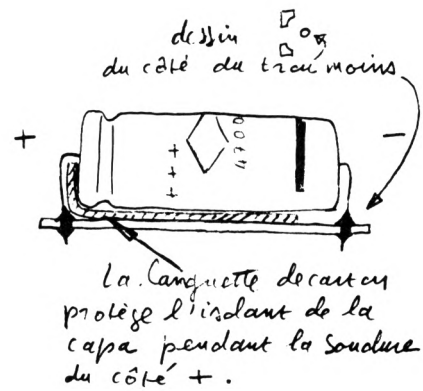
Soudez au revers de la carte en tenant les diodes perpendiculaires à la carte. Coupez les pattes à 1 ou 2 mm. Ecartez les diodes en éventail ensuite pour le refroidissement.



— Attaquons maintenant les quatre grosses capas de 4 700 uF 16 V. Toutes les quatre ont le + tourné à gauche (le + est imprimé sur la capa, c'est le bout avec une gorge rétreinte et un opercule noir isolant de fermeture).

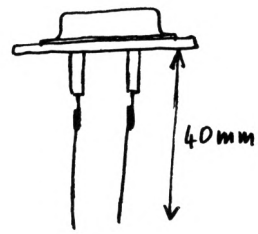
Attention de ne pas inverser, ces capas électrolytiques se transformeraient en pétards de gros calibres à la mise sous tension ! *Ecartez-en les diodes* à 5 mm au moins.

La dimension de la carte ayant été mesurée au plus juste, nous vous recommandons avant soudure de glisser sous la capa une languette de carton mince repliée comme indiqué. Soudez alors *des deux côtés de la carte* (ceci est valable également pour les capas de l'alimentation n° 1).

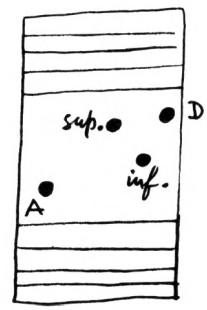


— Coupez sur du gros fil deux longueurs de 6 à 8 mm de gaine isolante, enfilez-les sur les pattes du régulateur LM 323 K et faites-les glisser jusqu'à la racine des pattes contre le boîtier.

Coupez et dénudez en entier deux longueurs de gros fil et soudez-les aux pattes du LM pour les rallonger jusqu'à 40 mm environ.

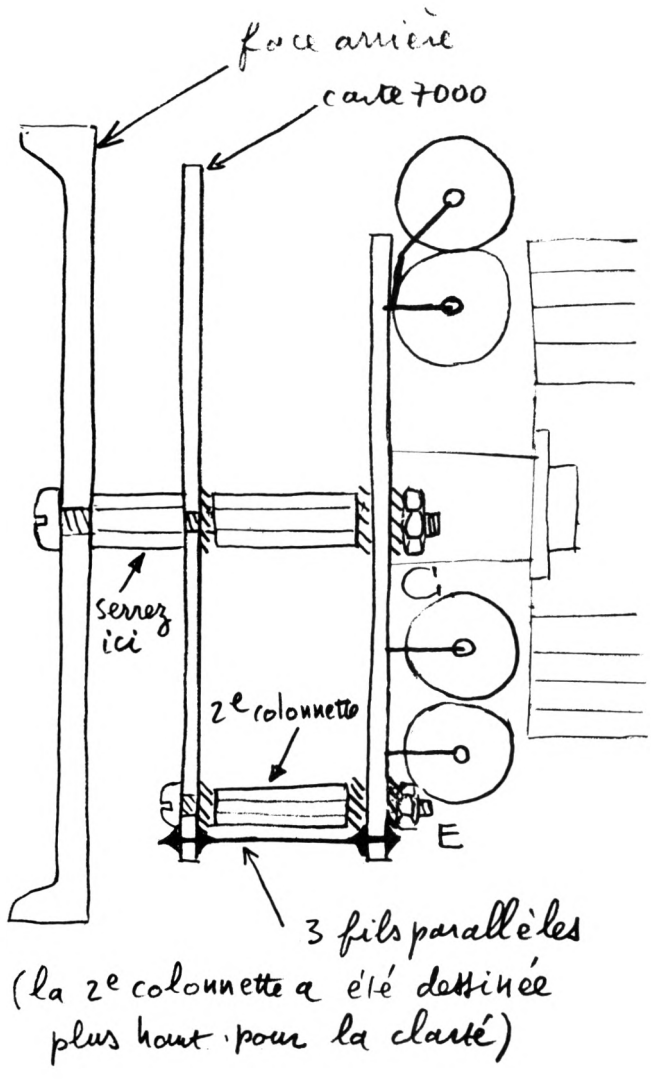


Prenez à la main le radiateur noir à 10 ailettes (ailettes sur le dessus) dans la position indiquée sur le croquis : les trous des pattes sont placés dissymétriquement, mettez-les à droite.



Enfilez par-dessus le LM, mettez à chacun de ses bouts une rondelle papillon dessus puis une vis de 3 x 12, et amenez l'ensemble au-dessus de la carte, c'est là que l'opération délicate commence : enfilez la patte supérieure du LM dans le trou de la carte qui est à gauche de D, et la patte inférieure dans le trou à gauche de B. On note une torsion en hélice des pattes vers la droite en descendant.

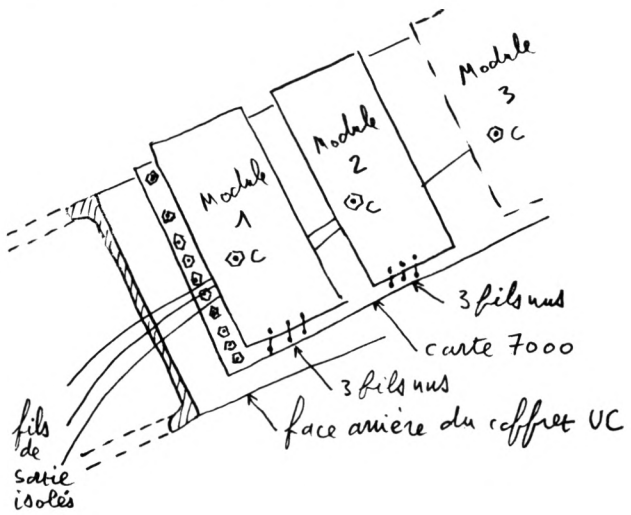
Vissez les vis sur les colonnettes A et D, soudez les pattes au revers de la carte, coupez-les, c'est maintenant presque fini.

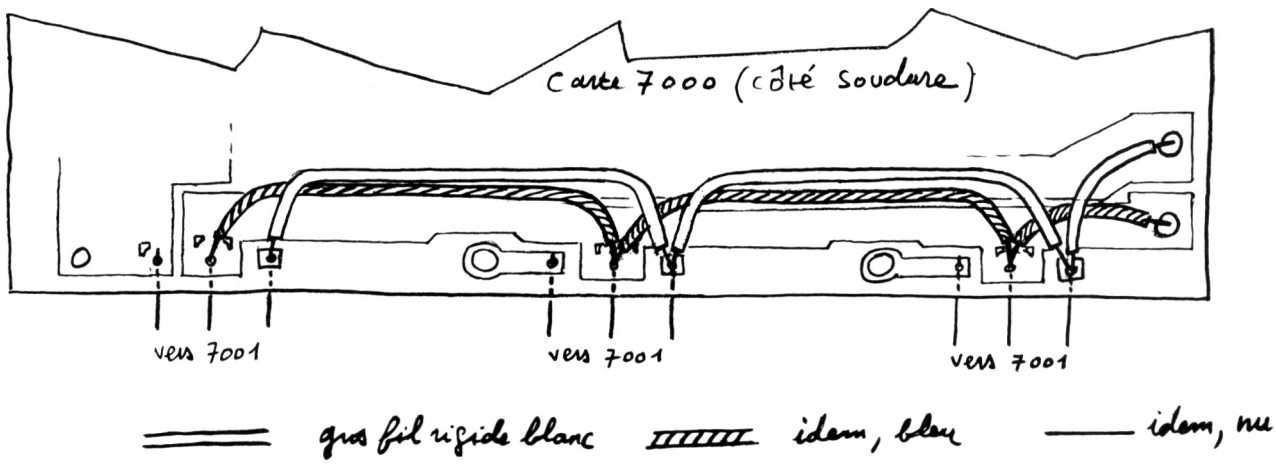


Montage sur la carte alimentation

Lorsque tous les modules sont prêts, démontez les 4 vis de la face arrière d'Alcyane pour fixer les modules sur la carte alimentation 7000.

Reliez ensuite les fils isolés des modules 5 V aux sections 5 V des cartes fond de panier, de façon à équilibrer au mieux les consommations.





Avant de monter les modules 7001, il faut ponter l'arrière de la carte 7000 en gros fil rigide comme indiqué sur le croquis.

7 - 4 - 2. Modules — 12 V

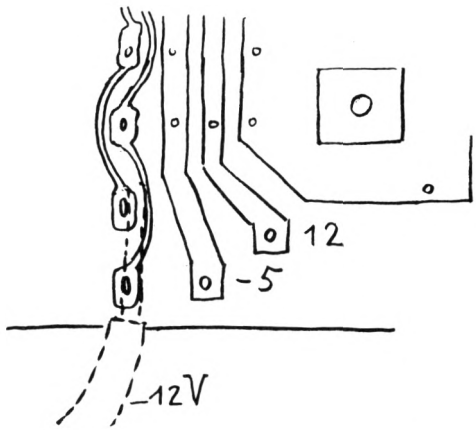
Le montage de ce module est plus simple, car le faible débit demandé par les ROM a permis de n'employer qu'une diode, en se contentant de redresser une seule alternance.

Remarquons à ce sujet que l'argument de simplicité évoqué par les fabricants de microprocesseurs demandant seulement du + 5 V pour leur alimentation est fallacieux : il ne tient que pour de très petits systèmes. Dès qu'on rentre dans le monde réel du traitement, avec des besoins en interfaces variés, une multiplicité de tensions devient nécessaire.

Prendre les précautions usuelles : gaînage des pattes du LM, méthode de mise en place de C1, respect général des polarités. Les composants sont placés sur l'époxy, soudure au dos.

Le fil — 12 V est à relier au fond de panier où sont montées les cartes nécessitant cette alimentation : la carte combinée et l'interface TV. Le passer par un trou de la cloison.

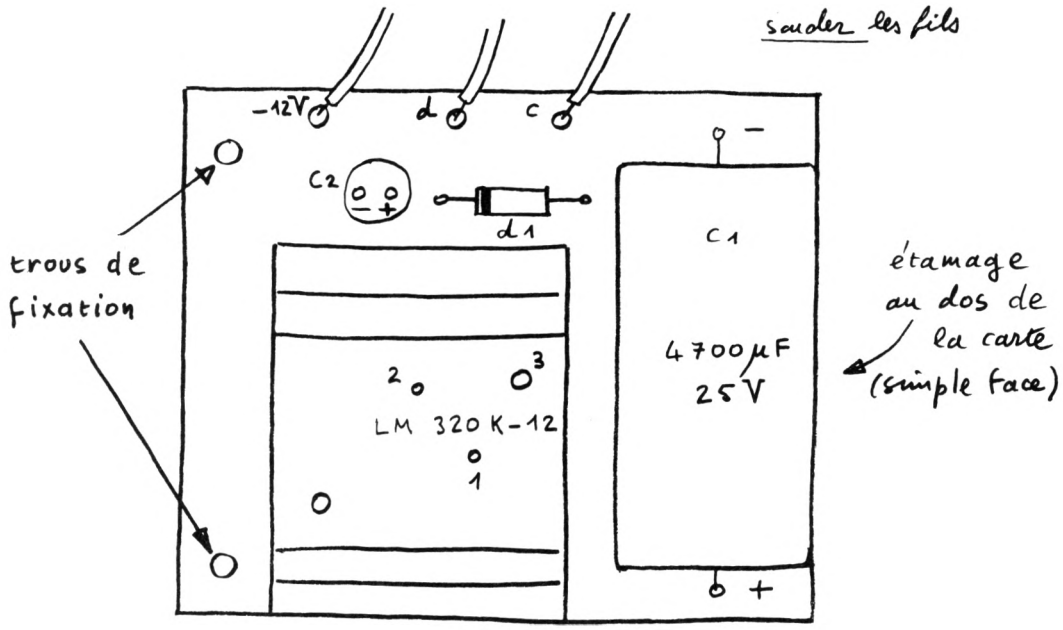
Démonter la carte fond de panier. Dénuder et étamer le fil souple sur 6 à 7 mm après l'avoir roulé entre les doigts pour être sûr qu'aucune barbule ne se produira. Le souder ensuite en recouvrement à plat sur la face métallisée de la carte fond de panier comme indiqué sur le croquis. Ou on peut le souder sur la quatrième paire de picots du connecteur le plus bas : choisissez la méthode qui vous paraîtra la moins difficile ! Ce qu'il faut, c'est alimenter en — 12 V les deux pistes qui longent la piste — 5.



FICHE TECHNIQUE ALIMENTATION

L'alimentation d'Alcyane est conçue de façon à pouvoir grandir avec le nombre de cartes installées dans le coffret.

Chaque carte a une consommation en 5V relativement importante. Il est donc nécessaire de moduler la puissance de l'alimentation selon la configuration choisie, et éventuellement de pouvoir l'augmenter ultérieurement.



Carte 7003
alimentation - 12V

Exemples de consommations (en ampères) :

carte CPU 1001	0,46 A
face avant	0,9 A
carte mémoire 4K	1,2 A
carte E/S	0,5 A
carte télévision	0,9 A
carte combinée	0,5 A

DESCRIPTION DES ALIMENTATIONS

ALIMENTATION DE BASE (N° 1)

Elle n'est utilisée que pour de petites configurations Alcyane.
Elle comprend sur une même carte :

- 1 module 5V 3A à 3,5A (MA),
- 1 module +12V 0,5A (utilisé principalement pour les cartes suivantes :

  CPU 8080 et PROM, TV, interface série CCITT V24 petite imprimante 20 colonnes),
- 1 module −5V 0,5A (utilisé par le CPU 8080 et la PROM),
- 1 transformateur 6A sur le +5V.

ALIMENTATION N° 2

Utilisée pour les configurations moyennes Alcyane. Elle comprend en plus de l'alimentation n° 1 :

- 1 module 5V 3A (MA1),
- 1 module −12V 0,5A (utilisé par : TV, petite imprimante 20 colonnes interface série CCITT V24),
- 1 ventilateur.

ALIMENTATION N° 3

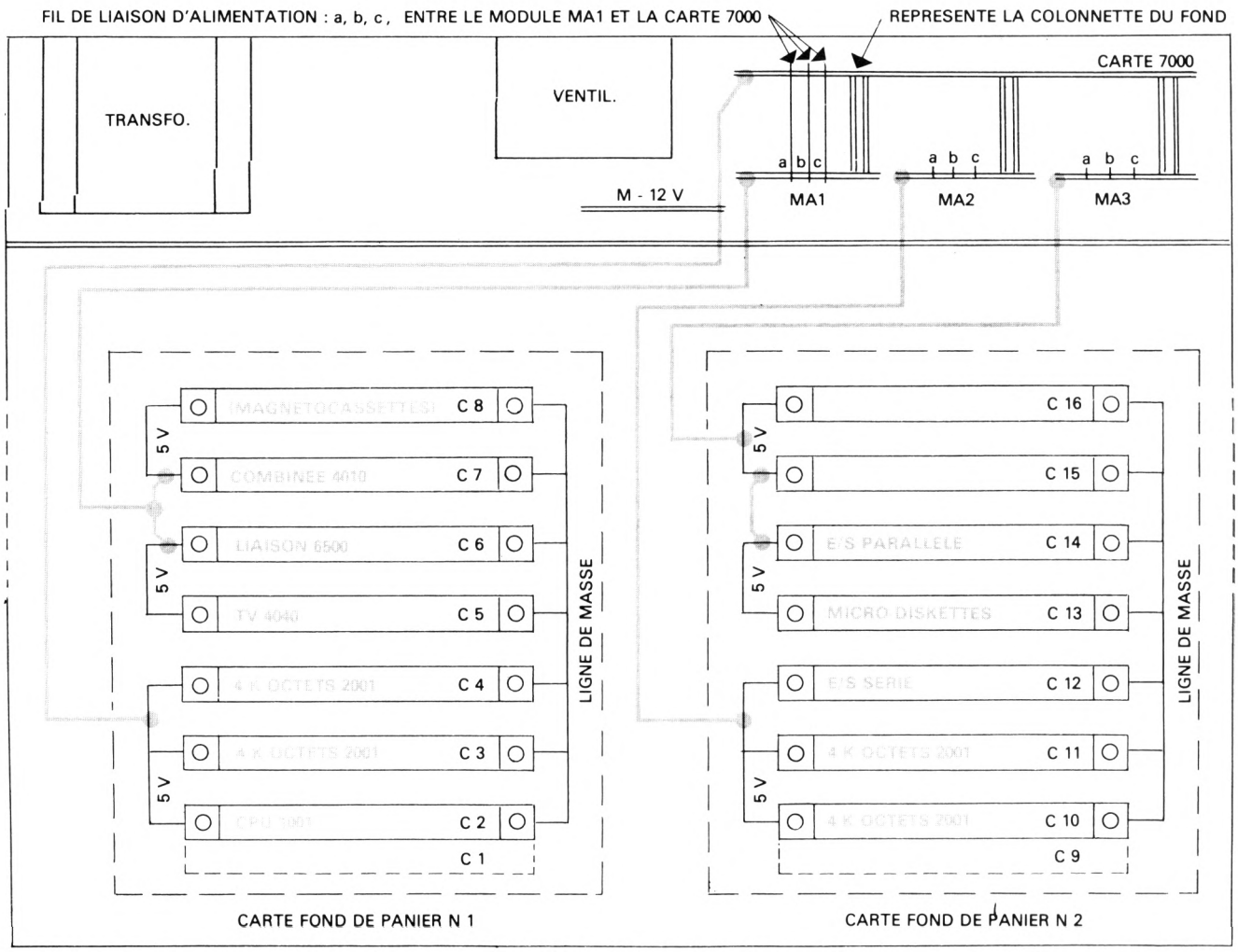
Elle comprend en plus de l'alimentation n° 2 :

- 2 modules 5V 3A (MA2 et MA3),
- 1 transformateur 5V 10A en remplacement du premier.

Chaque module d'alimentation 5V3A est réparti sur la carte fond de panier et alimente trois ou quatre connecteurs.

La somme des consommations sur un module doit être de l'ordre de 3A.

Nous avons reproduit à titre d'exemple la feuille d'équilibrage d'alimentation qui accompagne chaque livraison d'Alcyane : un double en est gardé dans le dossier de la machine pour mise à jour lors des extensions ultérieures.



7 - 5. INTERFACE TV

Il comprend deux grandes cartes video, la 4040 et la 4041, se montant au-dessus les unes des autres.

7 - 5 - 1. Les cartes vidéo

Carte 4040

Rappelons que de façon générale les pastilles où souder pattes de composants et câbles sont *carrées*. Les pastilles rondes sont les trous métallisés de passage d'une face à l'autre, et ne sont pas à souder. Enfoncer les tantales le plus bas possible.

Le montage de cette carte est facile. Postitionner les cavaliers aux adresses désirées si l'on programme en assembleur et qu'on a une application spéciale. Si non (impérativement pour le Basic) adopter les adresses standards suivantes :

— relier S1 à 24-31
— relier S2 à 4-5
— relier S3 à 2
— relier S4 à 0 (b)

De cette façon, l'adresse décimale du module TV sera 02, et le tampon interface de 1K sera situé de 28K à 29K.

De nombreux éléments des deux cartes vidéo sont prévus pour l'option graphique, et ne seront pas utilisés dans ce montage.

Carte 4041

C1 et C2 n'ont pas de polarité. Coucher à plat sur la carte tous les condensateurs pour diminuer l'encombrement, ce qui demande de leur laisser des pattes assez longues.

Les cavaliers sont tous à placer comme indiqué si l'on utilise un téléviseur aux normes françaises. S1 en DF inverse la polarité du signal de synchro, S3 en AB fait de même pour la vidéo.

Cette carte comporte une sortie vidéo directe pour moniteurs : souder dans ce cas l'âme du coax à la pastille V et la gaine à la pastille 0 (sur le bord de la carte). Il en est de même si votre TV est équipée d'une prise d'entrée directe vidéo (moyen le plus simple à utiliser).

Montage des cartes vidéo

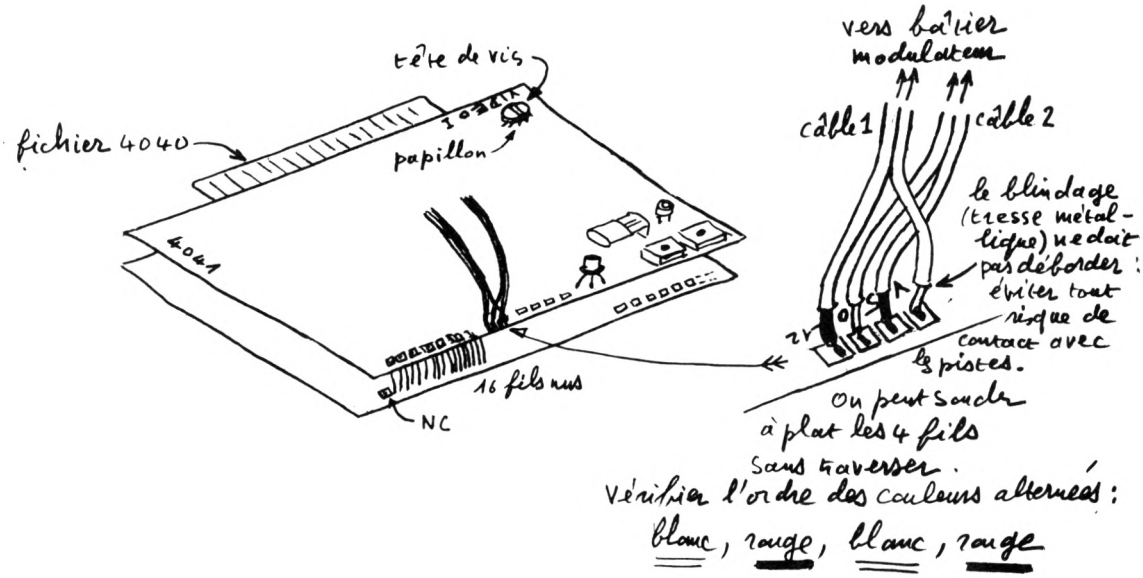
Placer la 4041 au-dessus de la 4040, les 16 pastilles carrées de chaque carte à l'angle en regard.

Monter la colonnette de l'angle (près de l'inscription VIDEO I) papillons dessus/dessous. Dénuder du fil rigide et en souder un premier tronçon à l'angle opposé en maintenant les deux cartes bien parallèles : à partir de là, elles sont maintenues et on soudera facilement les 15 autres fils. S'ils sont un peu longs, pincer les deux cartes entre les doigts pour les rapprocher : on a ainsi une marge de réglage.

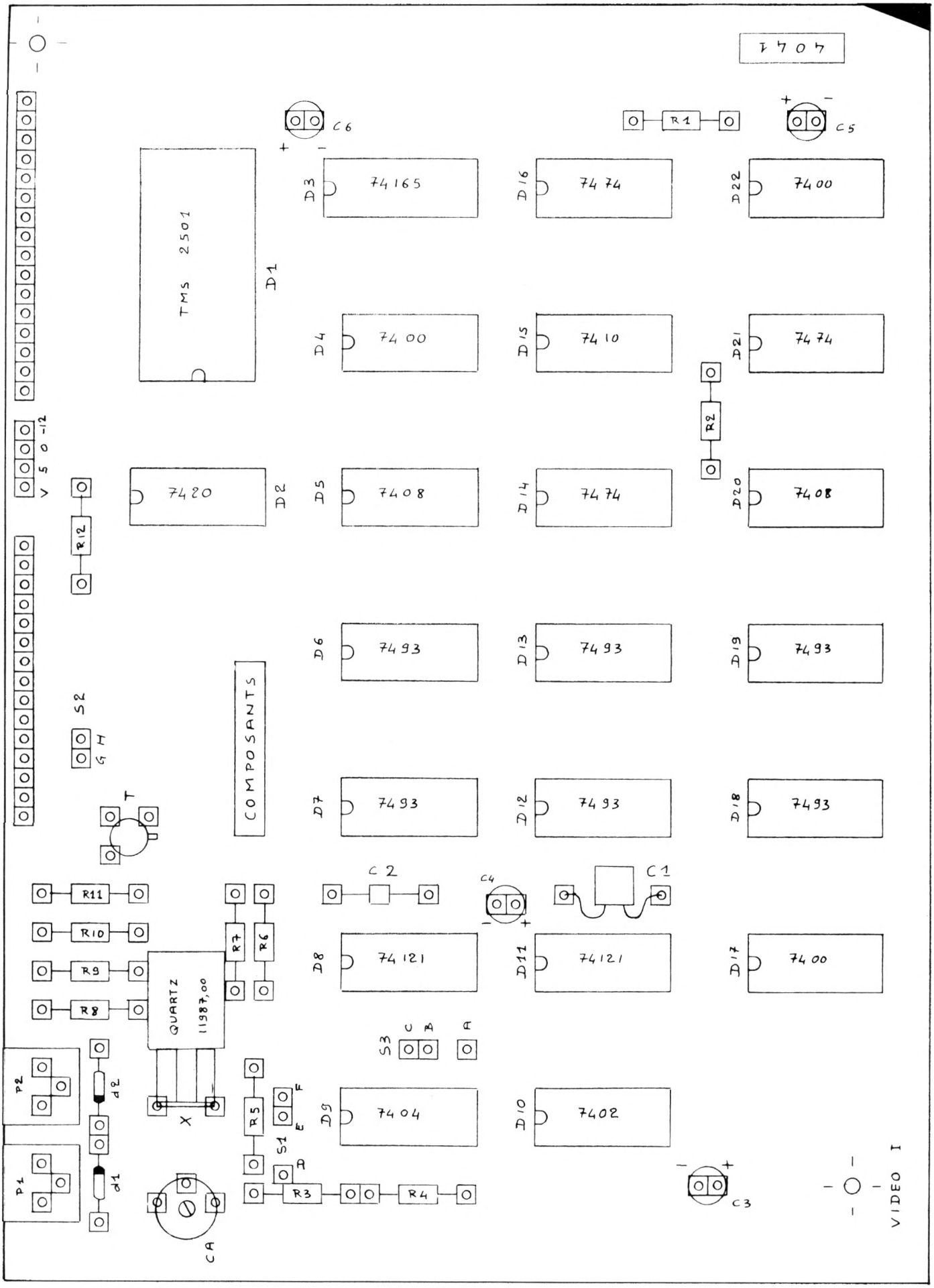
7 - 5 - 2. Utilisation et fonctionnement de l'interface télévision

Le but de cet interface est de visualiser, sur un téléviseur standard, deux pages de caractères alphanumériques, chaque page étant composée de 16 lignes de 32 caractères.

Les deux pages sont stockées dans un registre-tampon de 1 K Bytes, découpé pour des raisons d'adressage en quatre demi-pages de 8 x 32 caractères.



4010
COMPOSANTS
74157 D11
74157 D10
74157 D9
2102 D8
2102 D7
2102 D6
2102 D5
2102 D4
2102 D3
2102 D2
2102 D1
74126 D12
74126 D13
74126 D14
74175 D15
74 83 D16
74 93 D17
74 93 D18
74 93 D19
74 74 D20
7402 D21
7404 D22
74 20 D23
7442 D24
8205 D25
8205 D26
S1
S2
S3
S4
R1
R2
R3
R4
C1
C2
C3
FC1
FC2
NC


1707
C6
C5
R1
D3
74165
D16
7474
D22
7400
TMS 2501
D1
D4
7400
D15
7410
D21
7474
R2
V 5 0 -12
7420
D2
D5
7408
D14
7474
D20
7408
R12
D6
7493
D13
7493
D19
7493
S2
G H
COMPOSANTS
T
D7
7493
D12
7493
D18
7493
C2
C4
C1
R11
R10
R9
R8
R7
R6
D8
74121
D11
74121
D17
7400
QUARTZ
11987,00
S3
C
B
A
P2
d2
X
R5
S1
E
F
D9
7404
D10
7402
P1
d1
A
R3
R4
CA
C3
VIDEO I

Un numéro de demi-page stocké dans un registre "demi-page", et manipulé par un ordre IN/OUT, permet d'indiquer quelle demi-page se place en haut de l'écran (voir croquis).

7 - 5 - 3. Accès (timing, contrôles, codes)

Le chargement du buffer s'effectue comme pour n'importe quelle zone de la mémoire, sans délai. Deux hypothèses sont alors possibles :

a) chargement asynchrone : pendant tout le temps que dure l'accès à la mémoire, la visu est éteinte.

b) chargement synchrone :
le test préliminaire d'un drapeau permet de savoir si l'on se trouve pendant une période de "marge verticale" c'est-à-dire entre le haut de l'écran et le début du texte, ou entre la fin du texte et le bas de l'écran. Dans ce cas, on pourra écrire autant de caractères qu'on le désire, sans perturber l'image visualisée.

Le buffer est situable dans toute la zone-mémoire, par incréments de 2 K Bytes.
Une adresse IN/OUT définit :

a) le registre demi-page,
b) (IN) le drapeau.

(NB : elle est encodée en BCD).
L'adresse de l'interface est au choix, par cavalier. On la donne comme deuxième octet des instructions IN et OUT, le bit de gauche de cet octet étant égal à un (voir "Fonctionnement de la carte combinée"), ce qui donne par exemple OUT 82. On met préalablement dans l'accu l'adresse de la demi-page.

7 - 5 - 4. Génération de marges et de caractères, synchro

Le standard utilisé dérive du CCIR 625 lignes en ce sens que l'on visualise deux fois 312 lignes non entrelacées, 50 trames/seconde.

Carte génération de marges et de caractères

On part d'un quartz à 12 MHz moins 1 °/oo (un condensateur ajustable permet d'ajuster la fréquence à 50 x 312 x 768 Hz).

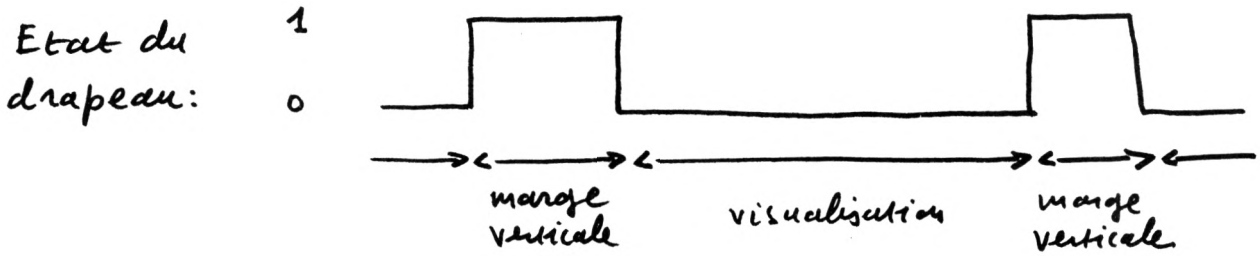
Une prédivision par deux fournit la fréquence de base :

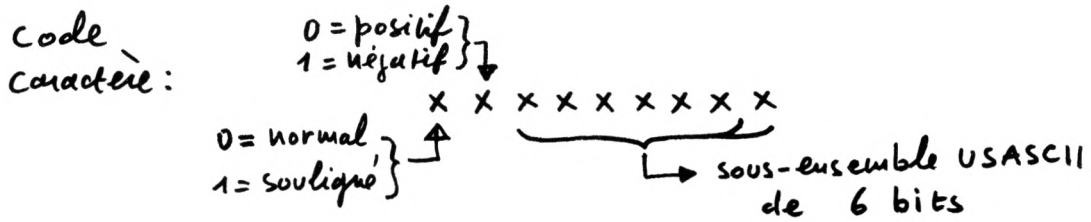
a) un compteur par 8 fournit la séquence de points
b) un compteur par 48 fournit la séquence des caractères,
c) un compteur par 12 fournit la séquence de lignes de points,
d) un compteur par 26 fournit la séquence de lignes de caractères.

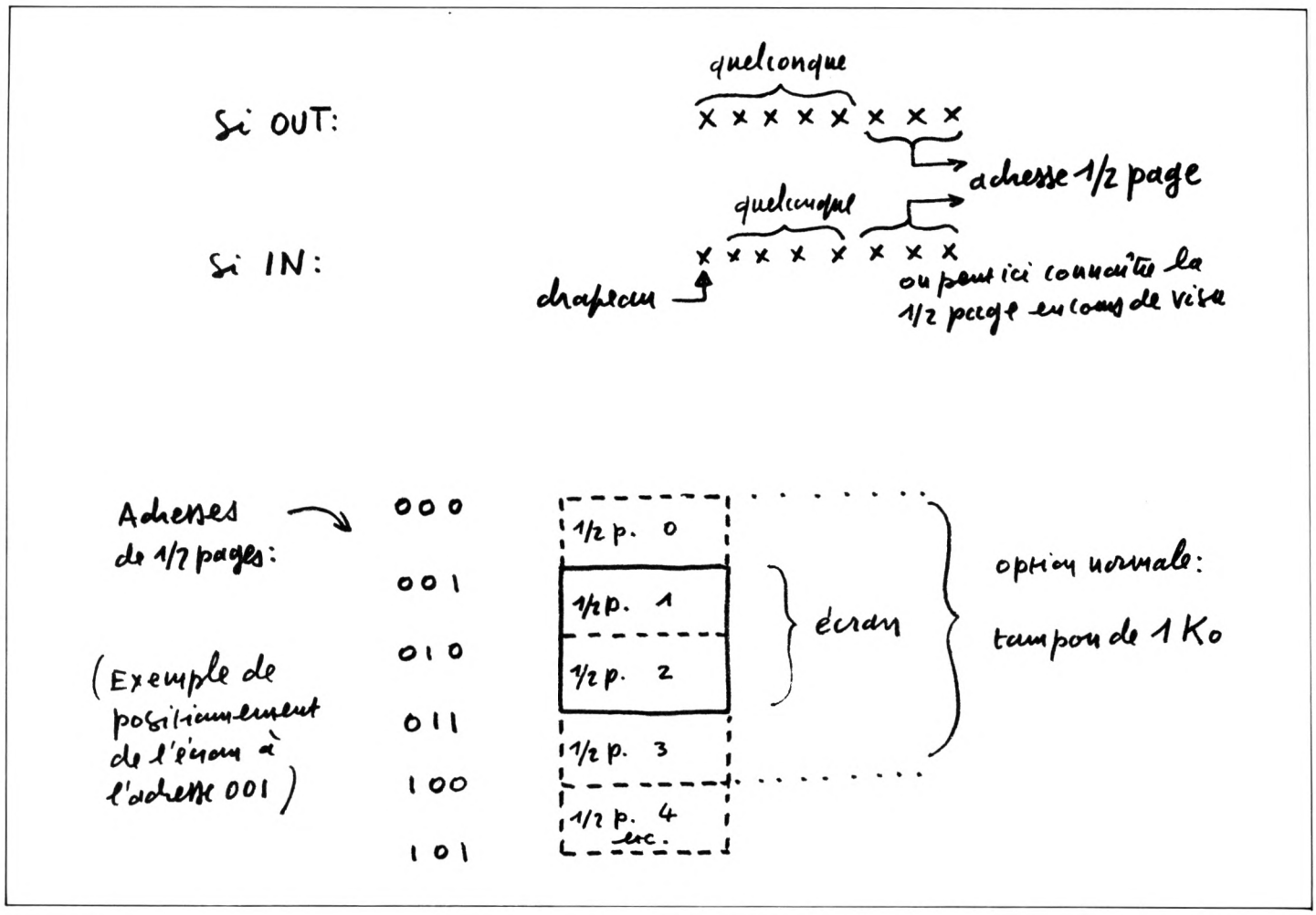
Génération des synchros :

— le passage à zéro du compteur (b) déclenche un monostable qui donne la synchro-lignes,

— le passage à zéro du compteur (d) déclenche un monostable qui donne la synchro-images.



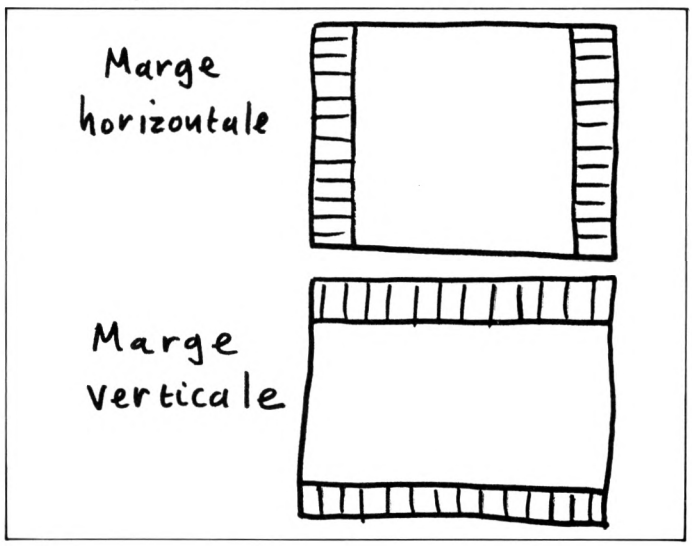




Génération des marges :

— un décodage sur le compteur (b) donne la marge horizontale,

— un décodage sur le compteur (d) donne la marge verticale.



Génération des caractères :

— quand le compteur (a) passe l'état 1, le 74165 (registre à décalage) est chargé avec les sorties du TMS 2501 (générateur de caractères en ROM), puis décalé pendant les sept autres coups du cycle du compteur (a) ; le signal série obtenu est (ou non) complémenté pour fournir le positif/négatif, puis haché à 6 MHz pour fournir le pointillé ; il est ensuite contrôlé par le produit des marges et envoyé au mélangeur/adapteur d'impédance (émetteur follower) ;

— le compteur (c), quant à lui, fournit les signaux d'exploration du TMS 2501, à savoir : sept lignes de caractères, une ligne de souligné, quatre lignes d'interlignes pendant lesquelles la vidéo est bloquée "au noir".

7 - 5 - 5. Carte de gestion mémoire

Le 74175 se charge et se lit par les ordres IN et OUT (adresse décimale déterminée par des straps) ; il contient l'adresse demi-page.

Quand on écrit dans la mémoire, le décodage anticipé de l'adresse valide, strobé par STSTB, met à zéro le 7474 qui :

a) bloque le vidéo au "noir",
b) commute les 74157 sur les lignes d'adresse (A0... A10),
c) valide les signaux Mem R et Mem W qui permettent alors de charger et de lire la mémoire.

A l'instruction d'après (au STSTB suivant) le bistable repasse à un, indiquant que la visu est de nouveau validée.

Dans ce mode, la vidéo est de nouveau valide, et le compteur de caractères (les trois 7493) est :

a) remis à zéro au début de chaque cycle (montée de READY = FIN DE MARGE VERTICALE),
b) incrémenté pour la section colonne à chaque impulsion CP qui a lieu en même temps que le chargement du 74165,
c) incrémenté pour la section ligne à chaque impulsion AVI qui a lieu en même temps que l'interligne.

Le rôle du 7483 est d'assurer la commutation demi-page du haut / demi-page du bas.

N.B. : la prévision d'une éventuelle option 4 pages alphanumériques / 1 page graphique 128 x 128 a rendu nécessaires :

— la sortie de nombreux fils normalement non utilisés,
— l'adressage de la zone mémoire par pas de 2 K bytes.

7 - 6. DEUXIEME FOND DE PANIER

Lorsque le quota de sept connecteurs du premier fond de panier est épuisé, il faut monter le deuxième fond de panier, prolongation interne du bus. On dispose ainsi de sept autres connecteurs banalisés (au-delà, on utilisera un deuxième coffret).

La liaison de ce fond de panier supplémentaire avec le premier s'effectue de deux façons différentes.

7 - 6 - 1. Solution do-it-yourself

C'est celle qui est illustrée en détail dans la planche page 182. Elle correspond à un coût minimum, soit : un circuit imprimé et quelques mètres de câble plat. Sans présenter de difficulté à proprement parler, le travail de soudage demande à être fait minutieusement et contrôlé régulièrement.

7 - 6 - 2. Solution de luxe

Elle utilise : le circuit imprimé fond de panier n° 2, deux connecteurs et un long circuit imprimé en U portant à chaque bout un fichier latéral. On soude un connecteur sur chacun des fonds de panier en position basse (position zéro du croquis chapitre 5 - 7, § i), et après remontage dans la machine on enfiche le circuit imprimé dans les deux connecteurs. C'est plus facile... mais les circuits imprimés de grande taille (et les connecteurs) sont chers.

7 - 7. INTERFACE MAGNETOCASSETTE

Cet interface a pour but de permettre aux utilisateurs d'Alcyane de stocker et de relire programmes et données, de façon économique, avec des magnétocassettes (MK7) courants non transformés employant des cassettes (K7) haute fidélité courantes.

C'est lui qui permet le chargement de software d'Alcyane (assembleur, Basic) lorsque la machine ne comprend pas de disques. L'interface MK7 est donc fondamental pour de tels systèmes. Le chapitre qui lui est consacré est découpé de la façon suivante :

— Pour ceux qui programment aux clefs (sans software) :
  1. description du mode d'enregistrement,
  2. utilisation aux clefs,
— Pour ceux qui utilisent le software d'Alcyane :
  3. utilisation avec la PROM MK7,
  4. utilisation sans la PROM,
  5. montage et schémas de l'interface,
  6. ordres Basic K7.

7 - 7 - 1. Description du mode d'enregistrement

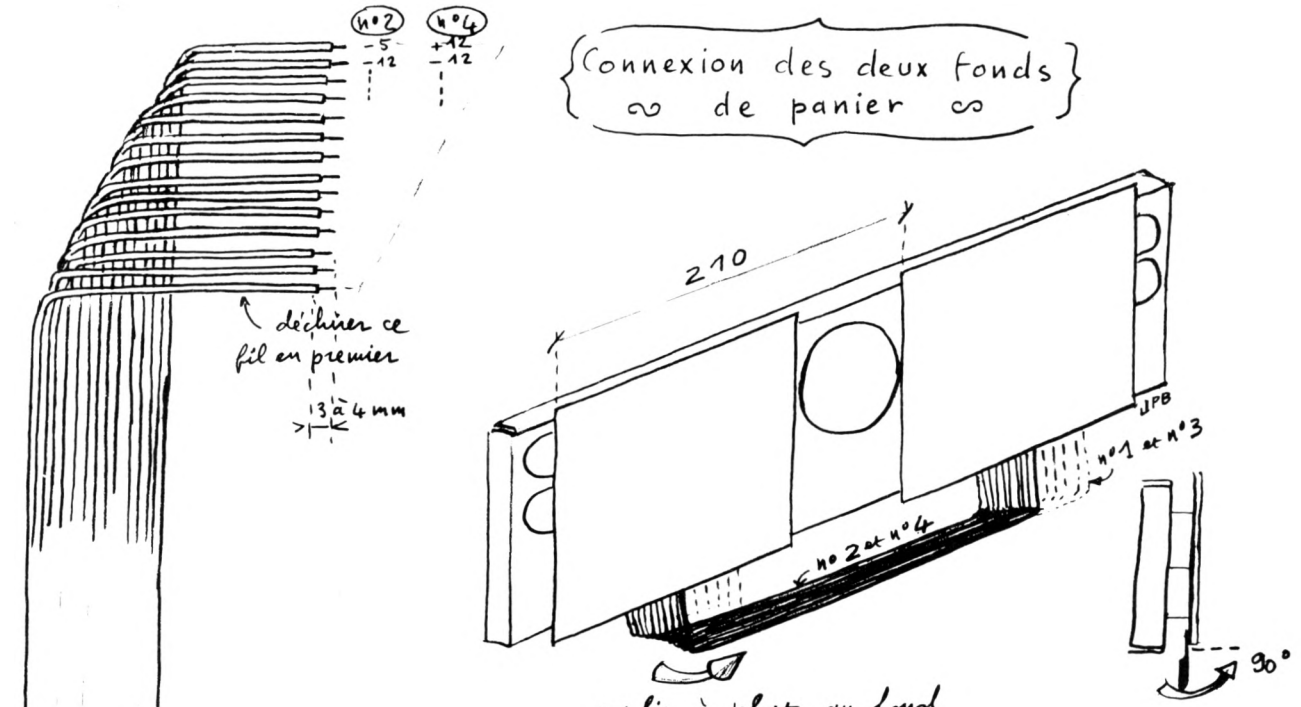
Une K7 est avant tout faite pour enregistrer des fréquences musicales (parole, musique), en gros de 100 à 10 000 Hz.

C'est conformément à cette vocation qu'elle est utilisée pour l'interface d'Alcyane. L'information est inscrite en bifréquence :

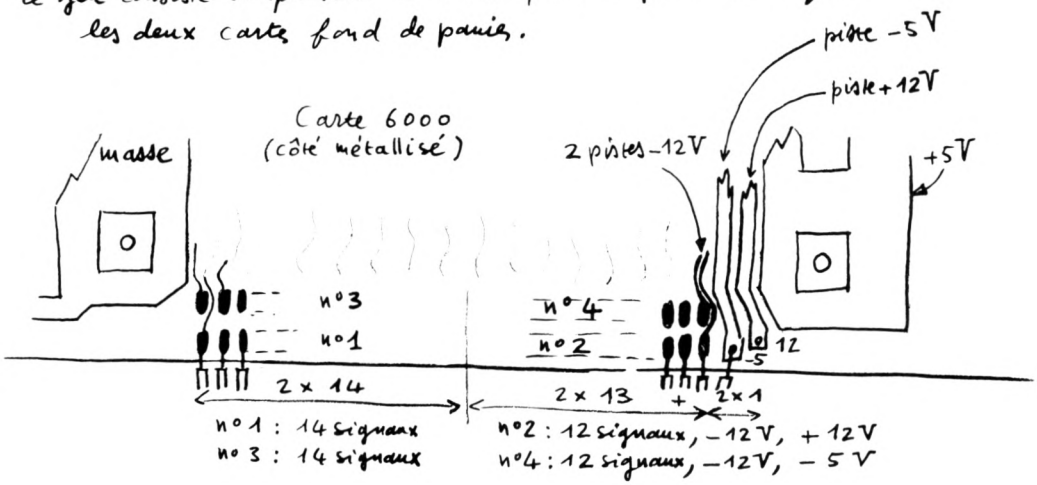
4651 Hz pour zéro
3846 Hz pour un

On entendra donc chanter l'information ! Les valeurs apparemment bizarres des fréquences correspondent à une période de 215 microsecondes pour zéro, et une période de 20 % plus longue environ pour un : 260 microsecondes. Elles résultent bien sûr de divers compromis techniques, et passent facilement dans tout magnétophone convenable : l'information est lue ou écrite sur la cassette en *mode asynchrone* à une vitesse de 414 bauds. Le baud, du nom de notre compatriote Baudot (Jean-Maurice-Emile, 1845-1903), inventeur du "télégraphe imprimeur" qui porta son nom, et

Connexion des deux fonds de panier



Le jeu consiste simplement à relier plot à plot identique les deux carte fond de panier.



14 fils

← gabarit, échelle 1/1 : partir de 33 cm de câble, séparer en premier ce fil, puis le suivant, etc.

couper ici. Tenir le fil à la pince plate pour le dénuder (sinon on risque de le séparer involontairement trop loin), ou bien le remettre parallèle à l'ensemble du câble pour dénuder.

Préparer 4 câbles identiques : n°1, n°2, n°3 et n°4.

Souder les fils à plat sur les plots après étamage. Attention aux barbules. La soudure n'apparaîtra pas dessous, les trous n'étant pas métallisés. Bien vérifier à chaque fois qu'on n'a pas croisé deux fils. Inspecter soigneusement, à la loupe si nécessaire.

CONSEILS DIVERS

Manipulation

Un conseil : les circuits imprimés qui ont rendu Alcyane possible sont un produit de haute technicité à manipuler avec un certain respect. Les pistes ont 35 microns d'épaisseur, et ne sont pas faites pour recevoir des chocs d'outils (causes de fractures difficiles à voir, d'où pannes rebelles).

Soudage

Il ne faut pas souder trop vite : plutôt deux secondes pour une patte de DIP normal, et beaucoup plus sur la carte d'alimentation, les connecteurs sur le fond de panier, l'embase du quartz, etc. Pour les plus grosses soudures, si l'on n'a pas de Weller régulé, un fer de 50 W minimum est nécessaire.

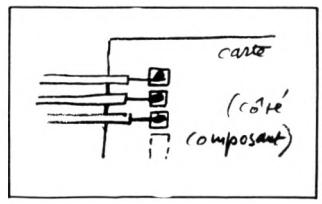
Un léger débordement de la soudure côté composant est sans inconvénient : en fait, il faut que la soudure remplisse bien le trou, et son apparition de l'autre côté indique que le trou est bien métallisé.

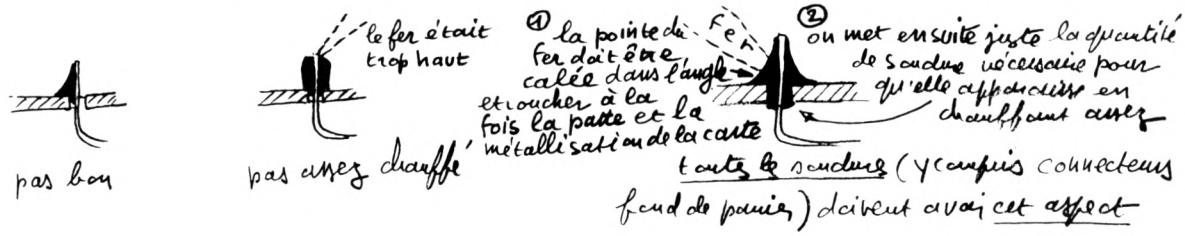
toute erreur (notre ou votre...), et les classer ou les étiqueter.

Certains condensateurs n'ont pas de polarité (condensateurs céramiques ou autres de faible valeur : petits carrés noirs, disques plats).

Soudage à plat

Il est plus facile de souder les câbles plats à plat sur les pastilles carrées du bout de la carte sans les passer par les trous. Attention aux barbules pouvant causer des courts-circuits. Vérifier que la soudure est apparue à la sortie du trou côté SOUDURE.





Précautions

Un Weller est assez bien isolé pour vous dispenser du fil de mise à la terre de la panne, mais touchez de temps en temps la feuille d'aluminium avec la panne.

Remarquez bien que sur certaines cartes il y a des pistes reliant entre elles des pattes adjacentes de DIP. Lorsque vous vérifiez votre travail, ne les grattez pas en les prenant pour des ponts de soudure !

Si vous avez à dessouder, fondez parfaitement la soudure avant de commencer à tirer, sinon vous risquez d'arracher une piste (dans ce cas, réparez avec un fil).

Composants discrets

Bien poser les résistances à plat au contact de la carte. Après soudure, couper les pattes *bien à ras* de la soudure. De même pour les condensateurs, LEDs, etc. *Ne pas crocheter* les pattes sous la carte (sinon le dessoudage devient impossible sans dégâts à la carte).

Vérifier soigneusement les codes de couleur des résistances avant de commencer le travail, pour éviter

Cavaliers nus (reflets dans un œil d'étain...)

Une longueur de fil blanc très mince est fournie pour y couper les cavaliers. Après les avoir coupés à la longueur, les dénuder *entièrement*, puis mettre en place et souder.

Quartz froid

Les supports de quartz ont une forte inertie thermique. Chauffer longuement la soudure fondue pour réussir la soudure.

PROM en zéro

Lorsqu'on reçoit une 8704/8, après l'avoir installée et avoir positionné le cavalier de la carte 1001, changer le cavalier de la carte mémoire 4 K qui était à l'adresse zéro et le mettre à la première zone 4 K libre contiguë aux autres cartes mémoire.

qui est aujourd'hui le telex, est l'unité de vitesse de transmission de l'information. Nous contentant d'une définition proche de la vraie, réservée elle aux puristes, nous dirons que le baud représente une vitesse de un bit par seconde. Notre cassette débitera donc 414 bits/seconde, ce qui donne une durée de 2415 microsecondes pour chaque bit. Si c'est un 0 il sera représenté par 11 périodes de sinusoïde successives, un 1 par 9 périodes : on voit que ce moyen d'inscription est très sûr.

La sécurité est encore améliorée par la réalisation électronique de l'interface à l'aide de *PPL (phase-lock loops, boucles à verrouillage de phase)*. Un mot s'impose au passage à propos de ces circuits : développés dans le cadre de la technologie radar, ils cherchent un signal périodique noyé dans le bruit, et se verrouillent dessus. Si le signal devient très faible, ou même disparaît carrément pendant un temps de l'ordre d'une période, le PPL reste verrouillé dessus : il se comporte comme un volant en mécanique.

Les PLL sont maintenant disponibles sous la forme de circuits intégrés : il y en a deux par interface, un pour l'écriture, un pour la lecture. Sur l'une des pattes de ce dernier nous aurons un état 0 ou 1 selon la fréquence présente sur la bande. A partir de maintenant, nous parlerons surtout des états des PLL, ce sont eux qui nous intéressent en programmation, et pas les fréquences.

En mode asynchrone, chaque caractère comprend :

— un bit start (bit de départ)
— les bits d'information
— un ou deux bits stop.

Tout ceci est entièrement piloté par le programme driver de l'interface, qui détermine par boucle d'attente la durée des bits, effectue la prise de synchronisation, etc., comme nous allons le voir.

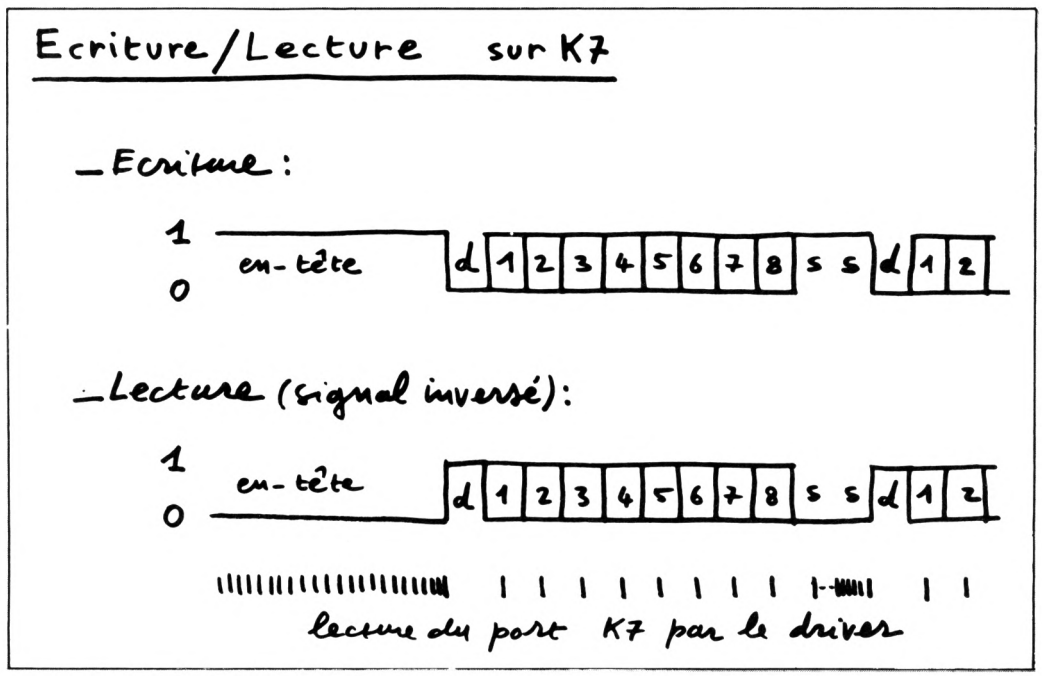
On voit sur la planche qu'en attente d'un caractère on est à 1. Puis arrive le bit de départ, qui est un zéro. Ensuite les huit bits de l'octet, puis deux bits stop qui sont des 1. Au total donc, 11 bits par octet. Et on recommence par un bit zéro qui indique l'octet suivant, etc.

Ce qui est asynchrone, c'est l'arrivée des caractères : ils peuvent être plus ou moins espacés les uns des autres, puisque leurs bits d'encadrement les rendent indépendants. Dans la pratique, ici, les caractères provenant de la mémoire centrale (et non d'une frappe à vitesse variée sur un clavier) seront juxtaposés par le driver, un bit start suivant immédiatement le deuxième bit stop.

A l'intérieur du caractère, et dès le bit start, le cadencement des bits est parfaitement régulier (d'autant plus que la vitesse de déroulement des instructions du driver est régulée par quartz, celui qui pilote le microprocesseur) : on est synchrone au niveau du bit.

Le stop a été fixé par programme à la durée de deux bits (comme il n'y a pas de changement d'état, qui reste à 1 pendant tout le stop, ou aurait pu prendre le temps de 1,5 bit par exemple, ou toute autre valeur) de façon à laisser tout le temps de se livrer par programme à la manipulation des octets successifs, ainsi qu'aux contrôles qu'on verra plus loin.

Rappelons au passage qu'en transmissions à 5 moments (c'est-à-dire 5 bits d'information par caractère, le code Baudot) le stop dure le temps de 1,52 bit... La durée des créneaux sur les matériels



électromécaniques est déterminée par des cames ouvrant et fermant des contacts : le temps du stop devait être 1,5 bit, mais le dessinateur se trompa en calculant l'angle de la came... et le standard international est devenu et resté 1,52 !

A la lecture de notre K7, les niveaux se trouvent inversés par les circuits électroniques : l'en-tête sera donc à zéro, le start à 1 et le stop à zéro. Attention en programmation... Pour récupérer les caractères, on fait une prise de synchro qui se déroule de la façon suivante :

a) lecture répétée par une boucle rapide du port (*) du MK7 : on "échantillonne" zéro des milliers de fois,
b) brusquement, on lit 1, c'est le flanc montant du créneau start. A partir de là, on tourne une boucle d'attente de 1,5 bit qui nous amène au milieu du premier bit d'information : la synchro est prise,
c) lecture du bit d'information, attente de un bit, lecture du bit suivant, attente de un bit, etc. jusqu'au bout,
d) on se trouve alors au milieu du premier bit stop : là, on se livre aux travaux de ménage nécessaires, puis on se met à échantillonner des zéros à grande vitesse... c'est a) en attente de l'octet suivant.

(*) Pour l'anglais "port", qui désigne un "orifice" par où passe l'information, nous avons conservé le mot français "port", qui signifie aussi "point de passage" : col du Somport, St-Jean-pied-de-port, en bas du col, etc.

Programme de lecture

La planche suivante indique l'ordre d'écriture des bits sur la bande, au départ de l'accu par l'instruction RLC.

L'autre donne la méthode employée pour le programme de lecture (les boucles d'attente et le comptage des bits successifs dans le registre C ne figurent pas).

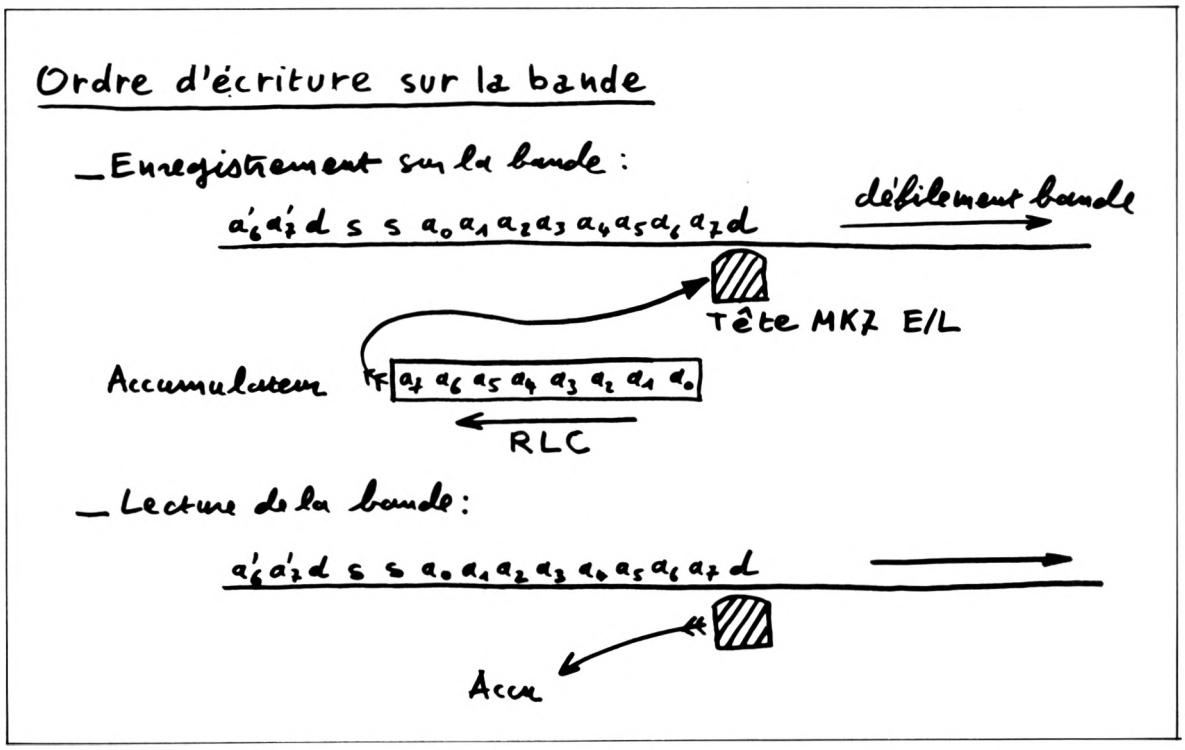
En se référant au schéma de la carte combinée, on voit que le bit lu ou écrit sur MK7-1 ou MK7-2 est DB_7, c'est-à-dire le bit le plus à gauche de l'octet : c'est lui que nous allons récupérer par un tour de passe-passe utilisant le drapeau de report CY.

Au départ on met neuf dans C, quant au contenu de B il est quelconque.

L'instruction IN nous amène en position gauche dans l'accu A le bit a_7 ainsi que sept autres bits sans objet.

RAL gare a_7 dans CY ; MOV A,B amène dans A le contenu initial (quelconque) de B ; RAL le fait circuler d'un cran à gauche en amenant a_7 dans la position la plus à droite de A ; enfin MOV B,A range le contenu de A dans B.

Le coup d'après on lit a_6, et par le jeu des décalages il vient se juxtaposer à droite de a_7. De proche en proche les bits progressent, jusqu'à l'obtention de l'octet complet dans A et B au dernier tour de boucle.



Programme de lecture

Etat de C			CY	Acc	B
9	IN		x	a_7 x x x x x x x	
8	RAL		a_7	x x x x x x x x	
	MOV	A,B	a_7	x x x x x x x x	
	RAL		x	x x x x x x x a_7	
	MOV	B,A			x x x x x x x a_7
8	IN		x	a_6 x x x x x x x	
7	RAL		a_6	x x x x x x x x	
	MOV	A,B	a_6	x x x x x x x a_7	
	RAL		x	x x x x x x a_7 a_6	
	MOV	B,A			x x x x x x a_7 a_6
7	IN		x	a_5 x x x x x x x	
6	RAL		a_5	x x x x x x x x	
	MOV	A,B	a_5	x x x x x x a_7 a_6	
	RAL		x	x x x x x a_7 a_6 a_5	
	MOV	B,A			x x x x x a_7 a_6 a_5
6	IN		x	a_4 x x x x x x x	
5	RAL		a_4	x x x x x x x x	
	MOV	A,B	a_4	x x x x x a_7 a_6 a_5	
	RAL		x	x x x x a_7 a_6 a_5 a_4	
	MOV	B,A			x x x x a_7 a_6 a_5 a_4
5	IN		x	a_3 x x x x x x x	
4	RAL		a_3	x x x x x x x x	
	MOV	A,B	a_3	x x x x a_7 a_6 a_5 a_4	
	RAL		x	x x x a_7 a_6 a_5 a_4 a_3	
	MOV	B,A			x x x a_7 a_6 a_5 a_4 a_3
4	IN		x	a_2 x x x x x x x	
3	RAL		a_2	x x x x x x x x	
	MOV	A,B	a_2	x x x a_7 a_6 a_5 a_4 a_3	
	RAL		x	x x a_7 a_6 a_5 a_4 a_3 a_2	
	MOV	B,A			x x a_7 a_6 a_5 a_4 a_3 a_2
3	IN		x	a_1 x x x x x x x	
2	RAL		a_1	x x x x x x x x	
	MOV	A,B	a_1	x x a_7 a_6 a_5 a_4 a_3 a_2	
	RAL		x	x a_7 a_6 a_5 a_4 a_3 a_2 a_1	
	MOV	B,A			x a_7 a_6 a_5 a_4 a_3 a_2 a_1
2	IN		x	a_0 x x x x x x x	
1	RAL		a_0	x x x x x x x x	
	MOV	A,B	a_0	x a_7 a_6 a_5 a_4 a_3 a_2 a_1	
	RAL		x	a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	
	MOV	B,A			a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0

7 - 7 - 2. Le programme de lecture

Pour l'utilisateur qui travaille sans software, nous avons prévu un programme destiné à l'utilisation d'un magnétophone par commande directe au pupitre d'Alcyane (face avant). Il permet une recopie intégrale du contenu de la mémoire sur K7, ainsi que sa relecture ultérieure. Afin de rester simple, ce programme a été réalisé sans périphériques de dialogue tels que télévision et miniclavier. De même, l'interface magnétophone comprend une commande par programme de démarrage/arrêt moteur : elle n'est pas utilisée ici.

Le programme normal de gestion d'un ou plusieurs magnétophones, ou d'une taille mémoire plus importante, utilise obligatoirement le miniclavier et un poste de télévision. Il est livré, lui, dans une PROM 8704 ou sur une cassette de simulation PROM : nous en reparlerons plus loin.

1. Implantation en mémoire

Ce programme doit être situé à la fin d'un module 4 Ko, ou en fin de mémoire.

L'adresse de la carte combinée, sur laquelle est fixée la carte magnétophone, doit être 00, carte magnétophone n° 1.

2. Cassette "RECOPIE MEMOIRE"

En travail normal, à la mise sous tension d'Alcyane, il n'est nécessaire d'introduire qu'une partie du programme : cette partie, dessinéé à la lecture K7, est le programme H002-1 (identique au H002, mais sans commentaires). L'autre partie, H001, qui sert à l'écriture sur K7, sera conservée sur une cassette qui ne contiendra qu'elle. Il est donc, au départ, nécessaire de fabriquer cette cassette "RECOPIE MEMOIRE", et de la mettre ensuite soigneusement de côté, puisqu'elle sera utilisée très souvent.

Alors que le programme H002-1 est donc à introduire aux clefs à chaque mise sous tension d'Alcyane (si, bien sûr, on envisage d'utiliser le magnétophone), le programme H001 n'est à introduire aux clefs qu'une seule fois, pour la création de la cassette "RECOPIE MEMOIRE".

En effet, H001 est capable d'écrire le contenu de la mémoire sur K7 : après l'avoir entré aux clefs, si on l'exécute il s'écrira donc sur cassette *lui-même*, en même temps que toute la mémoire. Cette possibilité peut sembler fort étrange au premier abord, et vous demandera sans doute de bien réfléchir pour voir comment l'opération peut s'exécuter sans interférence avec le fonctionnement du programme.

L'ensemble contigu des programmes (H001 lu, et H002 entré aux clefs) est implanté en fin d'un module 4 Ko ou, plus généralement, en fin de mémoire. L'emplacement choisi se détermine en remplaçant la lettre x par la valeur donnée par le tableau suivant :

Place en mémoire des programmes H001, H002	Valeur de x correspondante	Taille mémoire recopiée sur la cassette
fin du 1er module de 4 Ko	0	4 Ko
fin du 2e module de 4 Ko	1	8 Ko
fin du 3e module de 4 Ko	2	12 Ko
fin du 4e module de 4 Ko	3	16 Ko
⋮	⋮	⋮

L'enregistrement (ou la lecture) d'un bloc de 4 Ko prend environ une minute trois quarts.

Nota : la partie du programme H001 donnée en fin de liste (17 octets) ne suit pas la règle ci-dessus énoncée et reste implantée aux adresses données en début de mémoire.

Création de la cassette "RECOPIE MEMOIRE"

Insérer la prise SIGNAL dans le magnétophone.
En position MANUEL faire :

— RAZ,
— mode CLEFS, INSCR,
— introduire intégralement le programme H001,
— le relire soigneusement en EXAM.

Passer en mode PROG (toujours en position MANUEL), faire :

— RAZ,
— introduire une cassette vierge dans le lecteur, s'assurer qu'elle est bien rembobinée,
— passer en mode AUTO,
— immédiatement après, démarrer le magnétophone en position "enregistrement".

Le programme sera complètement enregistré lorsque tous les voyants de la face avant seront allumés sans clignoter (arrêt sur un HALT) :

— arrêter le magnétophone quelque dix secondes après,
— passer en mode MANUEL,
— faire RAZ, c'est fini.

Il est possible de contrôler sommairement que tout s'est bien passé en écoutant la bande. Au tout début, vous devez entendre un son continu assez aigu (de l'ordre de 4000 Hz). Ceci correspond à l'en-tête de l'enregistrement et permettra au programme de lecture de se synchroniser. Vous devez ensuite entendre un son un peu plus grave et crépitant, qui correspond au contenu de

la mémoire. En dernier lieu, le son redevient identique à celui de l'en-tête : l'écriture du programme est finie.

Utilisation de la cassette "RECOPIE MEMOIRE"

La lecture de cette cassette détruit presque complètement le contenu de la mémoire, il est donc nécessaire de l'effectuer dès la mise sous tension d'Alcyane si l'on envisage d'utiliser un magnétophone (*pour lire* une cassette aussi bien que pour écrire une cassette).

Suivre les instructions données sur la feuille Prog. Réf. H002-1 pour introduire le programme de lecture de la cassette "RECOPIE MEMOIRE".

Cette opération n'est à faire qu'une seule fois en début de travail.

a) Pour recopier le contenu de la mémoire sur cassette :

— introduire en mode CLEFS (position MANUEL) :
C3 JMP
80 adresse du
xF programme écriture
— passer en mode PROG,
— RAZ,
— passer en AUTO,
— démarrer immédiatement le magnétophone *en enregistrement.*

La recopie est terminée. Lorsque tous les voyants de la face avant s'allument (arrêt sur HALT) :

— passer en MANUEL,
— faire RAZ,
— arrêter le magnétophone.

b) Pour lire le contenu d'une cassette et le mettre en mémoire :

— introduire en mode CLEFS, position MANUEL
C3 JMP
D0 adresse du
xF programme lecture,
— passer en mode PROG,
— RAZ,
— démarrer le magnétophone *en lecture,*
— passer en AUTO quelques secondes après l'écoute du son d'en-tête.

La lecture de la cassette est terminée losque tous les voyants de la face avant s'allument (arrêt sur HALT) :

— passer en MANUEL,
— faire RAZ,
— arrêter le magnétophone.

REMARQUES

— Lors d'une lecture ou écriture d'une cassette, l'ordre des opérations MANUEL/AUTO et démarrage magnétophone est à respecter scrupuleusement. Sinon, le programme de lecture minicassette ne pourrait pas de synchroniser sur les informations provenant du magnétophone,

— il est possible, avec un peu d'habitude, d'enregistrer le contenu de la mémoire n'importe où sur une cassette, et non uniquement à son début.

Pour cela, il faut repérer sur le compteur du magnétophone le début de l'enregistrement (correspondant au début du son caractéristique de l'en-tête) lors de l'écriture.

Recaler ensuite grossièrement, lors de la lecture, la bande un peu avant le début d'en-tête, puis passer du mode MANUEL à AUTO quelques secondes après l'écoute de l'en-tête.

Ces quelques secondes d'attente évitent une éventuelle synchronisation sur des bruits magnétiques correspondant au démarrage moteur effectué lors de l'enregistrement.

— Les trois premières instructions sont utilisées à l'écriture d'un JMP pour branchement et exécution des programmes lecture ou écriture. Il est donc absolument nécessaire de réserver ces trois emplacements par des NOP ou, ce qui est identique, par C3, 03, 00.

— Une partie du programme H001 est écrite en haut de mémoire (à partir de l'adresse 0000 jusqu'à 0010). Ses instructions ne sont exécutées qu'une fois, à la fin de la lecture de la cassette RECOPIE MEMOIRE. Elles peuvent donc être détruites par la suite, ne servant plus à rien.

— A l'usage de ceux qui veulent écrire leurs propres programmes de K7, exprimenter, etc.

- l'adresse du premier MK7 est CØ, celle du deuxième est AØ. On peut en brancher d'autres en rajoutant des cartes combinées et cartes interfaces (voir le montage).
- le bit D7 (gauche) de l'octet qu'on met dans l'accu pour envoi vers un MK7 est, comme vu, le bit d'information. Le bit D6 est la commande moteur : Ø pour la mise en route, 1 pour l'arrêt. On utilise ces deux bits simultanément. Par exemple, le démarrage du moteur et l'inscription d'un en-tête de 1 se fait par :

```
MVI    A,8Ø
D3     CØ
```

suivi d'une boucle d'attente de la longueur désirée pour se positionner ultérieurement au compteur et/ou à l'oreille, soit 5 à 20 secondes. *Si par contre on écrit une succession*

RECOPIE MÉMOIRE MINICASSETTE " PROGRAMME ÉCRITURE "

	Instructions		adresses	code instructions		commentaires
	LXI	SP	xFØØ	31		— initialisation pointeur de pile
	—		1	CC		
	—		2	xF	*	
	MVI	C	3	ØE		— initialisation boucle d'attente
	FF		4	FF		pour création d'en-tête
	MVI	D	5	16		
	2E		6	2E		
→	MVI	A	7	3E		
	ØØ		8	ØØ		
	CALL		9	CD		— émission en-tête à 1
	—		A	B8		
	—		B	xF	*	
	DCR	C	C	ØD		
	JNZ		D	C2		
	—		E	85		
└	—		F	xF	*	
	LXI	H	xFØØ	21		— initialisation espace écriture
	—		1	C9		de la mémoire sur MK7
	Esp +1		2	xF	*	
→	DCX	H	3	2B		— mise à jour compteur mot
	MVI	D	4	16		— émission bit START
	Ø2		5	Ø2		
	XRA	A	6	AF		
	CALL		7	CD		— écriture bit
①	—		8	B8		
	—		9	xF	*	
	MOV	B,M	A	46		— chargement en B du mot à écrire
	MVI	C	B	ØE		— compteur bit du mot à écrire
	Ø8		C	Ø8		
→	MOV	A,B	D	78		
	CMA		E	2F		— complément du bit à émettre
	MVI	D	F	16		
	Ø2		xFAØ	Ø2		
	CALL		1	CD		— écriture du bit
	—		2	B8		
	—		3	xF	*	
	MOV	A,B	4	78		
	RLC		5	Ø7		— nouveau bit à écrire
	MOV	B,A	6	47		
	DCR	C	7	ØD		
	JNZ		8	C2		
	—		9	9D		
└	—		A	xF	*	
	MVI	D	B	16		— émission 2 bits STOP
	4		C	Ø4		
	MVI	A	D	3E		
	ØØ		E	ØØ		
	CALL		F	CD		— écriture des 2 bits

RECOPIE MÉMOIRE MINICASSETTE " PROGRAMME ÉCRITURE " *(suite)*

①	instructions		adresses	code instructions		commentaires
	—		xFBØ	B8		
	—		1	xF	*	
	MOV	A,H	2	7C		— détection fin d'espace mémoire
	ORA	L	3	B5		à écrire
	JNZ		4	C2		
	—		5	93		
	—		6	xF	*	
	HLT		7	76		— arrêt du programme
	ANI		8	E6		— sous-programme bit
	ØØ		9	ØØ		
	ORI		A	F6		
	ØØ		B	ØØ		
	OUT		C	D3		— écriture du bit
	ad.Magnéto		D	CØ	**	
	MVI	E	E	1E		— gestion boucle d'attente d'écriture
	7B		F	7B		du bit défini par
	DCR	E	xFCØ	1D		E = 1/2 bit
	JNZ		1	C2		D = multiple de E
	—		2	CØ		
	—		3	xF	*	
	DCR	D	4	15		
	JNZ		5	C2		
	—		6	BE		
	—		7	xF	*	
	RET		8	C9		— retour au programme appelant
	JMP		ØØØØ	C3		— saut au prog. écriture
	ØØ		1	ØØ		
	xF		2	xF	*	
	MVI	A	3	3E		— initialisation espace écriture et
	ØØ		4	ØØ		lecture par appel du programme
	STA		5	32		lecture
	91		6	91		
	xF		7	xF	*	
	STA		8	32		
	D1		9	D1		
	xF		A	xF	*	
	MVI	A	B	3E		
	76		C	76		
	STA		D	32		
	FD		E	FD		
	xF		F	xF	*	
	HLT		ØØ1Ø	76		PROG REF H001

RECOPIE MÉMOIRE MINICASSETTE " PROGRAMME LECTURE "

instructions		adresses	code instructions		commentaires
LXI	H	xFDØ	21		— chargement en H,L espace mémoire
—		1	C9		lu sur MK7 de (H,L) à ØØØØ
Espace +1		2	xF	*	
DCX	H	3	2B		— nouveau mot à lire
MVI	C	4	ØE		— Initialisation compteur de bits
9		5	Ø9		du mot à lire
IN		6	DB		— détection bit START
ad.Magnéto		7	CØ	**	
RLC		8	Ø7		
JNC		9	D2		
—		A	D6		
—		B	xF	*	
MVI	D	C	16		— initialisation compteur boucle
x + 1/2 x		D	Ø3		d'attente recherche du 1er bit
MVI	E	E	1E		du mot à lire
77		F	77		— initialisation compteur boucle
DCR	E	cFEØ	1D		d'attente d'une valeur de 1/2
JNZ		1	C2		bit
—		2	EØ		boucle d'attente pour échantillon
—		3	xF	*	du bit à lire
DCR	D	4	15		
JNZ		5	C2		
—		6	DE		
—		7	xF	*	
IN		8	DB		— lecture du bit
ad.Magnéto		9	CØ	**	
DCR	C	A	ØD		
JZ		B	CA		— saut rangement mot si C = Ø
—		C	F7		
—		D	xF	*	
RAL		E	17		— rangement en B du bit lu
MOV	A,B	F	78		
RAL		xFFØ	17		
MOV	B,A	1	47		
MVI	D	2	16		— initialisation compteur boucle
Ø2		3	Ø2		d'attente par recherche bit suivant
JMP		4	C3		— saut lecture d'un nouveau bit
—		5	DE		
—		6	xF	*	
MOV	M,B	7	7Ø		— rangement en mémoire du mot lu
MOV	A,H	8	7C		
ORA	L	9	B5		
JNZ		A	C2		— saut lecture d'un nouveau mot
—		B	D3		
—		C	xF	*	
JMP		D	C3		— pour la 1re lecture du prog écriture
ØØ		E	Ø3		saut au programme d'initialisation
ØØ		F	ØØ		et remplacement de cette instruction
					par HLT

PROG REF H002

PROGRAMME DE LECTURE BOOTSTRAP

ADRESSE	CODE INSTRUCTION
0000	C3
0001	00
0002	02
0200	21
0201	00
0202	02
0203	2B
0204	0E
0205	09
0206	DB
0207	C0
0208	07
0209	D2
020A	06
020B	02
020C	16
020D	03
020E	1E
020F	77
0210	1D
0211	C2
0212	10
0213	02
0214	15
0215	C2
0216	0E
0217	02
0218	DB
0219	C0
021A	0D
021B	CA
021C	27
021D	02
021E	17
021F	78
0220	17
0221	47
0222	16
0223	02
0224	C3
0225	0E
0226	02
0227	70
0228	70
0229	B5
022A	C2
022B	03
022C	02
022D	C3
022E	D3
022F	01

CODE HEXADÉCIMAL

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Pour la lisibilité,
les zéros
ne sont pas barrés
sur ce listage
sans alpha.

S 0001
= = = = = = =

RECOPIE MÉMOIRE MINICASSETTE

MODE D'EMPLOI CONDENSÉ

1. RAZ et MANUEL

2. introduire en mode CLEFS
   le programme suivant

3. Passer en mode PROG (position manuel)

4. Introduire la MK7 " RECOPIE MÉMOIRE "
   dans le lecteur

5. Mettre en route le magnétophone

6. A l'écoute de l'en-tête (bruit continu
   assez aigu) passer après quelques
   secondes en mode AUTO

Puis →

ÉCRITURE

En mode MANUEL + CLEFS
0000 C3
80
xF
puis en mode PROG, RAZ
et AUTO

LECTURE

En mode MANUEL/CLEFS
0000 C3
D0
xF
puis en mode PROG, RAZ
et AUTO

MODE D'EMPLOI
ÉCRITURE/LECTURE

0000	C3	
	D0	
	xF	*
xFD0	21	
1	C9	
2	xF	*
3	2B	
4	0E	
5	09	
6	DB	
7	C0	
8	07	
9	D2	
A	D6	
B	xF	*
C	16	
D	03	
E	1E	
F	77	
0	1D	
1	C2	
2	E0	
3	xF	*
4	15	
5	C2	
6	DE	
7	xF	*
8	DB	
9	C0	**
A	0D	
B	CA	
C	F7	
D	xF	*
E	17	
F	78	
0	17	
1	47	
2	16	
3	02	
4	C3	
5	DE	
6	xF	*
7	70	
8	7C	
9	B5	
A	C2	
B	D3	
C	xF	*
D	C3	
E	03	
F	00	

PROG REF H002-1

d'informations qui seront relues avec commande arrêt/marche du moteur par programme entre chaque bloc, un temps beaucoup plus court (de l'ordre de la seconde environ) suffit si moteur et régulation du MK7 sont de bonne qualité. Des essais vous montreront rapidement si vous accrochez la synchro à tout coup ; sinon rallonger le délai.

— Le choix des cassettes et le soin apporté à leur manipulation sont très importants. Ne pas les brutaliser avant/arrière. En cas de méli-mélo, ne pas forcer : démonter la K7 en dévissant les cinq vis, remettre en ordre et remonter, l'information sera fréquemment récupérable sans erreur. Comme pour tout support magnétique, prévoir des recopies de sécurité (directement d'un magnéto sur un autre, bien régler le niveau). Nous avons obtenu de bons résultats avec les K7 Agfa Superferro dynamic SM 60 (ou 60 + 6, de 10 % plus longues) que nous vous recommandons d'utiliser : leur qualité mécanique est correcte. Attention à la poussière au stockage. Resserrer légèrement les spires (inspection par le hublot) en tournant le bout du doigt dans un moyeu avant de mettre une K7 dans le MK7. L'état et le bon accrochage du presseur sont essentiels. Si la bande défile en biais dans la cassette, la lecture sera mauvaise. Nettoyer régulièrement les têtes et le cabestan du MK7 avec une K7 de nettoyage (à jeter au bout de 10 nettoyages) ou mieux avec un coton-tige imbibé d'alcool (isopropylique si possible) qui ne risque pas de rayer les têtes. Faites régler au départ et au moins tous les ans l'azimut de la tête et la vitesse du moteur. Nous recommandons les Philips/Radiola 2225 et 2219.

L'interface a beau rattraper beaucoup d'imperfections, un matériel à bas prix, sale, déréglé, et des cassettes gondolées à trois francs pièce, ne vous donneront qu'une piètre satisfaction ! Si tout est correct vous serez surpris de la fiabilité : aucun des interfaces K7 made in USA que nous avons essayés n'approche le nôtre d'une encâblure...

Utilisation avec la PROM MK7

Un programme moniteur cassette nettement plus élaboré — comportant en particulier des contrôles — est contenu dans la PROM magnétocassette. Il permet de charger du software (assembleur, Basic, programmes utilisateurs) dès la mise sous tension de la machine, sans avoir à entrer de programme aux clefs, avec un dialogue clavier TV.

La PROM formatte les informations sur la bande en blocs de 16 octets suivis d'un 17e octet de contrôle obtenu par OU exclusifs successifs des 16 précédents. Tous ces octets sont adjacents sur la bande : il n'y a pas de séparation entre blocs. Lors de la lecture, le programme refait au passage les OUX et compare avec l'octet de contrôle. S'il y a concordance, le compte de blocs lus affiché dans l'angle supérieur droit de l'écran TV est simplement augmenté d'une unité. Si un deuxième nombre s'inscrit en dessous, c'est qu'il y a une erreur de lecture : il faut recommencer l'opération, la poussière sera sans doute partie.

Mode d'emploi

En mode PROG, AUTO, dès la mise sous tension le RAZ automatique démarre le programme à l'adresse zéro, au début de la PROM.

Pour lire un programme système (assembleur, Basic, etc.) :

— appuyer sur la touche P du clavier : le mot PRET apparaîtra alors en haut à gauche de l'écran TV,
— mettre la cassette dans le magnétocassette et le positionner en lecture, puis appuyer sur la touche L du clavier : la K7 démarre, on entend passer l'en-tête, puis dès l'arrivée des informations le comptage des blocs apparaît.

Pour ré-initialiser un programme système déjà chargé :

— faire RAZ
— appuyer sur la touche G du clavier.

Remarque : utilisation au pupitre avec la PROM

Lorsqu'on travaille aux clefs, il est usuel :

— soit d'écrire les programmes à partir de l'adresse zéro, pour pouvoir les démarrer en faisant RAZ,
— soit d'entrer un JUMP dans les trois premiers octets pour aller, après un RAZ, vers l'adresse voulue.

L'une et l'autre de ces manœuvres deviennent impossibles quand la PROM est installée, puisque en zéro il y a du programme indélébile !

Rassurez-vous, ce problème a été envisagé, mais sa solution nécessite que nous nous plongions pendant quelques instants dans les premières instructions de la PROM qui sont :

Adresse hexa	Instruction (adresses hexa)
Ø	LXI SP
1	xx
2	xx
3	JMP
4	adresse routine

```
5       moniteur
.       .
.       .
.       .
8       LHLD
9       Ø2
A       1Ø
B       PCHL
.       .
.       .
.       .
10      LHLD
11      Ø4
12      1Ø
13      PCHL
.       .
.       .
.       .
38      LHLD
39      ØE
3A      1Ø
3B      PCHL
.       .
.       .
```

Lorsqu'on fait RAZ, on initialise le pointeur de pile puis on saute dans une routine du moniteur, qui appelle la routine clavier, laquelle attend qu'on appuie sur une touche, et restitue alors un caractère dans l'accu puis retourne à la routine moniteur. Celle-ci teste pour voir si c'est un P (affichage de PRET), un L (lecture cassette)... ou un G : c'est ce dernier qui nous intéresse. La reconnaissance du G provoque l'exécution de

```
LHLD    1ØØØH
PCHL
```

c'est-à-dire que l'octet en 1000 est mis dans L, celui en 1001 est mis dans H (ce sont les premiers de la mémoire vive), puis HL est envoyé dans le PC, réalisant ainsi un saut à l'adresse correspondante ; par conséquent, il vous suffit de mettre à cette adresse soit votre programme, soit un JMP qui vous envoie où vous voulez, et RAZ plus G vous y emmèneront directement à condition d'avoir mis ce qu'il faut en 1000 et 1001.

Pour les amateurs de programmation avancée, des possibilités supplémentaires existent. En utilisant le procédé ci-dessus, on fait exécuter une instruction EI pour armer les interruptions (qui à l'allumage sont désarmées par le RAZ, de même que par tout RAZ manuel ultérieur). A partir de là, une pression sur la clef INT au pupitre provoque un RST7, qui empile l'adresse de retour et exécute les instructions de la PROM à l'adresse du RST7, soit 38H : on exécute alors un saut à l'adresse contenue dans 1ØØF/1ØØE ce qui permet de varier les plaisirs...

Enfin, en cours de programme on peut provoquer des RST n variés, et selon n on exécutera les diverses séquences correspondantes dans la PROM pour de la voltige en souplesse.

Utilisation sans la PROM

Le moniteur cassette inscrit dans la PROM MK7 peut être livré enregistré sur K7. On le rentre alors en mémoire dans les adresses de 000 à 1FF, où il jouera exactement le rôle de la PROM absente.

Cette rentrée en mémoire se fait par un *bootstrap* (programme initial de lecture : bootstrap, en anglais, signifie lacet de chaussure, et en tirant assez fort sur le lacet... on peut se soulever soi-même en l'air !). Si vous préférez le français, vous pouvez dire *amorce*, mais l'humour s'en va.

Le bootstrap S 0001, qui fait 51 octets, est une version personnalisée de H 0002-1 uniquement destinée au chargement de la cassette "moniteur cassette". Nous en donnons le listage en hexadécimal seulement. Faites bien attention de positionner les clefs ADRESSE d'abord toutes à zéro, puis la clef A9 levée ensuite après les trois premiers octets pour l'inscription du reste du bootstrap. Relire le tout après inscription. Mettre la K7 rembobinée en place. Passer en PROG (toujours en MANUEL), faire RAZ. Démarrer alors le magnéto en lecture, et écouter la bande. Dès qu'on est bien dans le son pur de l'en-tête, après les couics variés du début, passer en AUTO. Le moniteur sera lu en mémoire centrale, après quoi ayant pris le contrôle il arrêtera le MK7. Si l'on veut rebobiner la K7, retirer légèrement la prise COMMANDE, puis la remettre après rembobinage.

Tout se passe maintenant comme si l'on avait la PROM (cf. le "Mode d'emploi" précédent). Cependant,, le soft sera nécessairement une version spéciale, puisque son adresse de changement en mémoire est après les 512 premiers octets de mémoire vive, et non à partir du premier octet disponible (à l'adresse 4K avec la PROM, cette dernière occupant l'espace d'adressage de 0 à 4K-1).

Montage et schémas de l'interface

La carte interface magnétophone est livrée normalement montée et réglée. Il ne reste plus qu'à la fixer sur la carte 4010 (combinée) et à souder le câble plat, ainsi que les câbles de raccordement au magnétophone :

— le câble blanc provenant du XR210 est à souder sur le plot + 12V de la carte 4010,
— le câble plat se soude dans l'ordre de marquage, à plat sur les plots,
— les parties sans blindage (tresses) des câbles blanc et rouge de raccordement doivent être d'une longueur minimum.

Nous vous demandons de ne pas toucher aux réglages, ils sont assez délicats. Il est impératif de vérifier que les straps de la carte de contrôle (5001) sont bien tous les deux en position basse (b et c reliés, a restant libre), sinon l'interface cassette ne fonctionnera pas, les boucles d'attente étant faussées.

Carte 9000 N° 1

Cette carte interface est réservée pour le magnétophone dit "Système". C'est par lui que sont lus les programmes de base tels que Assembleur ou Basic.

L'adresse de cette carte est automatiquement sélectionnée par le programme Bootstrap ou PROM minicassette.

Carte 9000 N° 2

Elle est destinée à relier un deuxième magnétophone utilisé comme périphérique normal de lecture/écriture.

Nota : le magnétophone N° 1 redevient, après lecture des programmes de base, un périphérique normal.

7 - 7 - 3. Utilisation des cassettes en Basic

Cinq instructions et une commande incorporées dans le Basic d'Alcyane permettent d'utiliser la cassette de façon très pratique.

Rappelons qu'une commande se frappe au clavier lorsqu'on est sous contrôle du moniteur (qui vient donc de placer son signe en début de ligne sur la TV, indiquant ainsi qu'il se tient attentif à votre service). La frappe se fait directement sans numéro de ligne, et dès qu'on appuie sur la touche RET, la commande est exécutée. Une commande ne peut être utilisée comme instruction dans un programme.

Une instruction fait normalement partie d'un programme, et est précédée d'un numéro de ligne, sauf lorsqu'on travaille en exécution immédiate sans mettre de numéro de ligne, auquel cas elle joue le rôle d'une commande. Si l'on met un numéro de ligne devant une commande, il est systématiquement ignoré, et la commande s'exécute dès la frappe du retour chariot.

Commençons donc par la commande K7 relative aux programmes :

Commande

LOAD [(x,y)]

Cette commande est destinée à lire un programme écrit sur une K7 : elle le charge en mémoire centrale, et pendant le chargement on voit apparaître en haut et à droite de l'écran TV un contrôle visuel des opérations, lorsque au sifflement pur de l'en-tête succède le grésillement des octets du programme. Le programme est en langage clair Basic : c'est d'ailleurs l'unique forme sous laquelle existe un programme Basic d'Alcyane, la forme source, *directement intelligible aussi bien à l'homme qu'à notre ordinateur muni de son "interpréteur" Basic,* ce qui fait la grande force de ce langage. Le programme est lu ligne par ligne (une ligne pouvant contenir plusieurs instructions) et l'on voit sur la TV le nombre de blocs de 16 octets nécessaire pour chaque ligne. Si une adresse apparaît sous cette indication, il y a eu détection d'une erreur de parité : arrêter le magnéto, rebobiner la cassette jusqu'au bon endroit pour recommencer la lecture, arrêter le magnéto, envoyer une interruption en abaissant une fois la clef INT sur la face avant d'Alcyane puis enclencher la touche start de lecture du magnéto, qui doit rester immobile.

Taper à nouveau au clavier (voir plus loin) et le chargement s'effectuera à nouveau, sans doute normalement.

Lorsqu'il y a deux magnétocassettes sur un système, il faut préciser le numéro de celui où a été placée la cassette programme. On écrira par exemple

LOAD (Ø,1)

indiquant de lire sur la carte combinée d'adresse Ø l'interface K7 d'adresse 1.

Ce numéro est optionnel, ainsi que l'indiquent les crochets (qui ne sont pas à frapper !!). En effet, de nombreux systèmes utilisent un seul magnétocassette, qui jouera alors le rôle de "périphérique système", avec l'adresse fixe 0,2. Il suffira alors de taper

LOAD

tout simplement.

Dès qu'on frappe la touche RET, le magnéto doit démarrer... si l'on n'a pas oublié d'appuyer au préalable sur sa touche de lecture ! Si c'est la cas, il suffit d'enfoncer ladite touche à ce moment-là, mieux vaut tard que jamais, et l'opération s'effectuera quand même.

Dès que le chargement du programme en mémoire centrale est terminé, le moniteur arrête le magnétocassette et envoie son signe en début de ligne : on est prêt à travailler. Si — bonne habitude — on désire alors rembobiner la cassette, surprise ! Rien à faire... et c'est normal, Alcyane a coupé le courant sur le magnéto...

Que faire ? La solution dans un instant... par la commande suivante. Mais nous avons un point plus important à voir d'abord : tout est clair si

avant d'opérer nous n'avions pas de programme déjà en mémoire. Et s'il y en avait déjà un ?

Dans ce cas, tout est affaire de numéros de ligne. Si le programme que nous chargeons n'a aucun numéro de ligne commun avec l'autre, il se placera devant ou derrière, dans le bon ordre, et nous exécuterons celui que nous voulons par la commande

RUN [numéro de ligne]

bien connue, qui ré-initialise toutes les variables.

S'il y a chevauchement, les lignes de l'ancien programme dont les numéros se retrouvent dans le nouveau seront détruites.

Si l'on désire, avant de faire LOAD, détruire l'ancien programme, on fait RAZ au pupitre d'Alcyane, puis on frappe la touche G du clavier : tout est ré-initialisé, et l'ancien programme disparaît de la mémoire. On est alors dans les mêmes conditions que lorsqu'on vient de finir de charger le Basic.

On voit donc que la commande LOAD s'utilise avec la commande RUN, toutes modifications du programme étant naturellement possibles avant de faire RUN.

Instruction 1

MOVE [(x,y)] expression

On retrouve une syntaxe identique à celle de la commande précédente en ce qui concerne le numéro cassette, et il en sera de même, d'ailleurs, pour les quatre autres instructions que nous verrons tout à l'heure : nous n'en parlerons donc plus.

Cette instruction met en route le magnéto, pendant un temps en secondes égal à la partie entière de l'expression, qui peut donc être un calcul mathématique, une combinaison de variables... ou plus simplement un nombre entier dans les cas usuels. En même temps que le magnéto est mis en route, un flot continu de zéros est envoyé sur la prise signal : si le magnéto est en position d'enregistrement, un en-tête, ou inter-bloc de la longueur indiquée sera inscrit. S'il est en position lecture, il se bornera bien sûr à défiler pendant le même temps.

MOVE permet donc en enregistrement de séparer des informations comme désiré, d'avancer la cassette à la lecture (par exemple pour sauter une zone perturbée), et aussi de résoudre notre problème de rembobinage. Après avoir exécuté notre LOAD, une fois le chargement terminé nous exécutons par exemple

MOVE 15

en le tapant sans numéro de ligne.

Que va-t-il se passer ? Le courant est rétabli pendant 15 secondes sur la cassette système : elle démarre en lecture, et de suite nous appuyons sur stop puis sur Rewind, elle se rembobine, et la fin de l'opération nous est signalée par le signal sonore ou le LED rouge du magnéto. Stop, puis éjection de la K7, soigneusement rangée dans sa boîte pour la prochaine utilisation. Naturellement, il faut mettre dans le MOVE une constante telle que le temps soit largement suffisant pour notre manœuvre, sinon la cassette n'ira pas jusqu'au bout, et il faudra retaper un deuxième MOVE, ce qui est sans inconvénient ; sauf celui de brutaliser la K7 programme par des à-coups successifs, à éviter. Le mieux est de mettre par exemple MOVE 9Ø et lorsqu'on a terminé de faire INT au pupitre pour en sortir.

Rappelons qu'avant de mettre une K7 dans le magnéto, il est bon de resserrer les spires en bloquant un moyeu avec un doigt et en tournant l'autre de même (pas trop fort !), pour minimiser les chances d'embrouillamini en enlevant le mou.

Instruction 2

LOAD [(x,y)] [numéro de ligne]

Cette instruction va lire un programme déjà stocké sur une K7, et le rentrer en mémoire centrle. La K7 doit être positionnée dans l'en-tête précédant le programme désiré (soit en début de bande, soit plus loin en s'aidant du compteur), comme pour la commande LOAD déjà vue. Le chargement du nouveau programme, vis-à-vis de celui qui est déjà en mémoire et qui contient l'instruction LOAD provoquant ledit chargement, se fait selon les numéros de ligne comme expliqué pour la commande LOAD.

Par contre, cette fois-ci, l'exécution du nouveau programme se déclenche toute seule par une commande CONTINUE implicite : les valeurs actuelles des variables sont donc conservées, ce qui permet d'enchaîner automatiquement des segments successifs d'un même programme. On continue au numéro de ligne indiqué dans LOAD, et au plus petit numéro si rien n'est spécifié. On voit les nombreuses possibilités de cette instruction :

— conservation d'un module principal en mémoire, qui appelle des modules successifs dans une même zone,
— écrasement intégral de chaque segment avec enchaînement (sans même avoir à spécifier de numéro de ligne, si tous commencent au même numéro),
— combinaison des deux méthodes précédentes, permettant l'exécution automatique de programmes d'une taille très supérieure à celle de la mémoire, avec la seule contrainte que l'ordre

des segments soit défini par avance (contrainte qui disparaît avec les diskettes).

Instruction 3

RECORD [(x,y)] [numéro de ligne], [numéro de ligne]

Cette instruction est utilisée pour enregistrer sur cassette un programme présentement en mémoire centrale. Si aucun numéro de ligne n'est indiqué, le programme sera écrit en entier sur la bande. Si un seul numéro figure, l'inscription se fera de cette ligne à la fin, et bien sûr s'il y a deux numéros, elle se fera du premier au deuxième numéro inclus. Avec une seule cassette et un programme à écrire en entier l'instruction sera donc simplement

RECORD

Avec deux cassettes et un programme à écrire de la ligne 17 à la ligne 242 on aura

RECORD (Ø,1) 17,242

Instruction 4

RECORD DATA [(x,y)] variable [,variable...]

Cette instruction nous permet d'écrire des données sur cassette, ces données étant des résultats de calcul, des données entrées au clavier, etc. On indique les noms des variables correspondantes, qui peuvent être aussi nombreuses qu'on le désire. Avec deux magnétos, et écriture sur le deuxième on aura par exemple :

RECORD DATA (Ø,2) C3, A, A1, B$, CØ, A3$

On précise donc une *liste de variables* dans cette instruction. A noter que les variables chaînes (ici B$ et A3$) peuvent être très longues (seule limite, la taille mémoire !), et qu'il n'est pas nécessaire de rappeler leur longueur.

De même, on peut écrire un tableau entier de façon très pratique comme l'indique l'exemple ci-après :

```
10  DIM        A (1Ø,1Ø), B$ (1ØØ)
                 •
                 •
                 •
25  RECORD      DATA  A(),  B$
                 •
                 •
```

Lors de la relecture, la liste des variables devra être redonnée dans le même ordre, sous la même forme que dans l'ordre d'écriture.

Outre ces utilisations en calcul, RECORD DATA est bien sûr l'instruction qui permet de constituer des fichiers, le programme constituant essentiellement en

•
•

```
                                 •
20   INPUT              XXXXXX
                                 •
                                 •
                validations/formattage/écriture
                                 •
                                 •
130  IF        A =   999999 THEN 140
135  GO         TO   20
140  DISP          "FIN DE TRAVAIL"
145  END
```

Instruction 5

LOAD DATA [(x,y)] variable [,variable...]

Cette instruction lit les valeurs d'une série de variables sur la cassette et affecte ces valeurs aux variables. La liste des variables doit être identique à celle de l'instruction RCORD DATA qui a permis de les enregistrer, et naturellement les variables de même nature et dimension (arithmétiques, chaînes, tableaux) mais pas nécessairement de même nom (encore que ce soit quand même plus pratique pour s'y retrouver dans les programmes !).

Pour résumer tout ceci, voici un commentaire détaillé du listage d'exécution reproduit ici.

Exemple d'utilisation

Le listage provient de l'imprimante rapide standard (80 colonnes, 200 caractères par seconde) qui est un excellent outil de travail.

La commande

PRINT ON

a été tapée au clavier dès le début : dès ce moment, tout le dialogue qui s'affiche à l'écran est imprimé au fur et à mesure.

Ce listage systématique ne s'arrêtera qu'en donnant la commande

PRINT OFF

Il est ainsi possible si on le désire de garder une trace écrite de toute la mise au point d'un programme, etc.

On voit donc s'imprimer le message signalant que le Basic (ici version 10 K) est chargé.

La commande

SEQUENCE 1Ø

indique que le premier numéro de ligne doit être 1Ø, le pas étant de 5 par défaut, puisque rien de plus n'est indiqué.

L'instruction

MOVE 1Ø

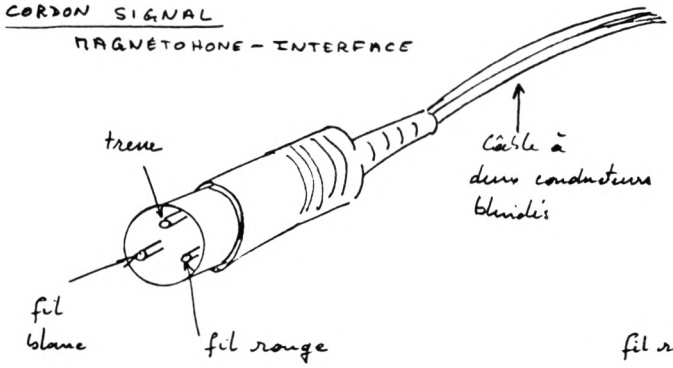
va écrire un en-tête de dix secondes, le magnéto ayant été au préalable enclenché en mode enregistrement.

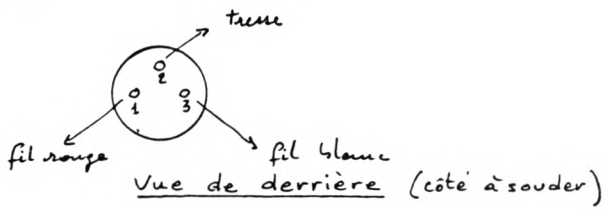
On entre dans une boucle FOR qui va faire se répéter dix fois les deux lignes qu'elle embrasse, où l'on calcule des variables A et B qu'on inscrit immédiatement sur cassette.

Le programme se termine par :

CORDON SIGNAL

MAGNÉTOPHONE – INTERFACE



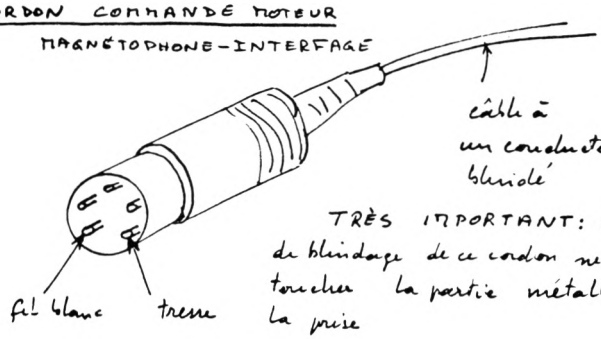


Vue de derrière (côté à souder)

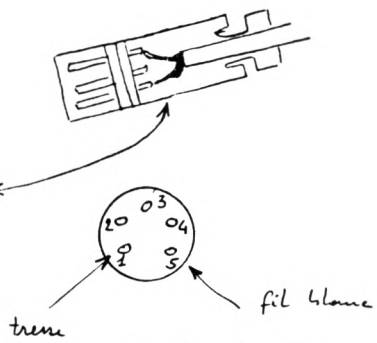
Les tresses de chacun des conducteurs doivent être soigneusement soudées à la fiche 2

CORDON COMMANDE MOTEUR

MAGNÉTOPHONE – INTERFACE

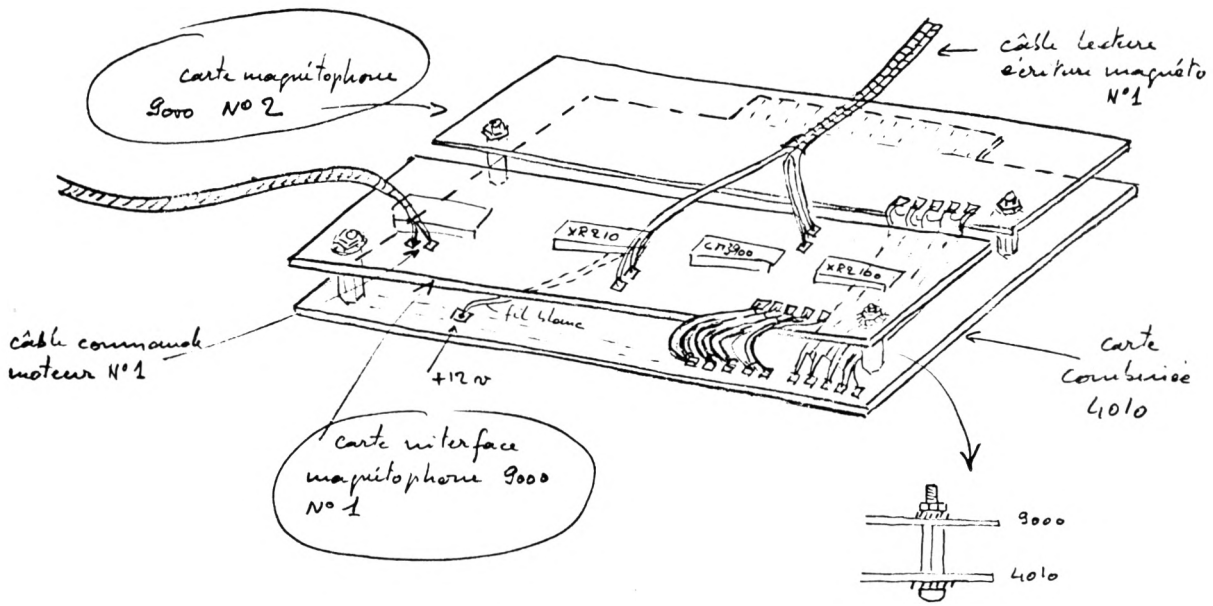


TRÈS IMPORTANT: la tresse de blindage de ce cordon ne doit pas toucher la partie métallique de la prise

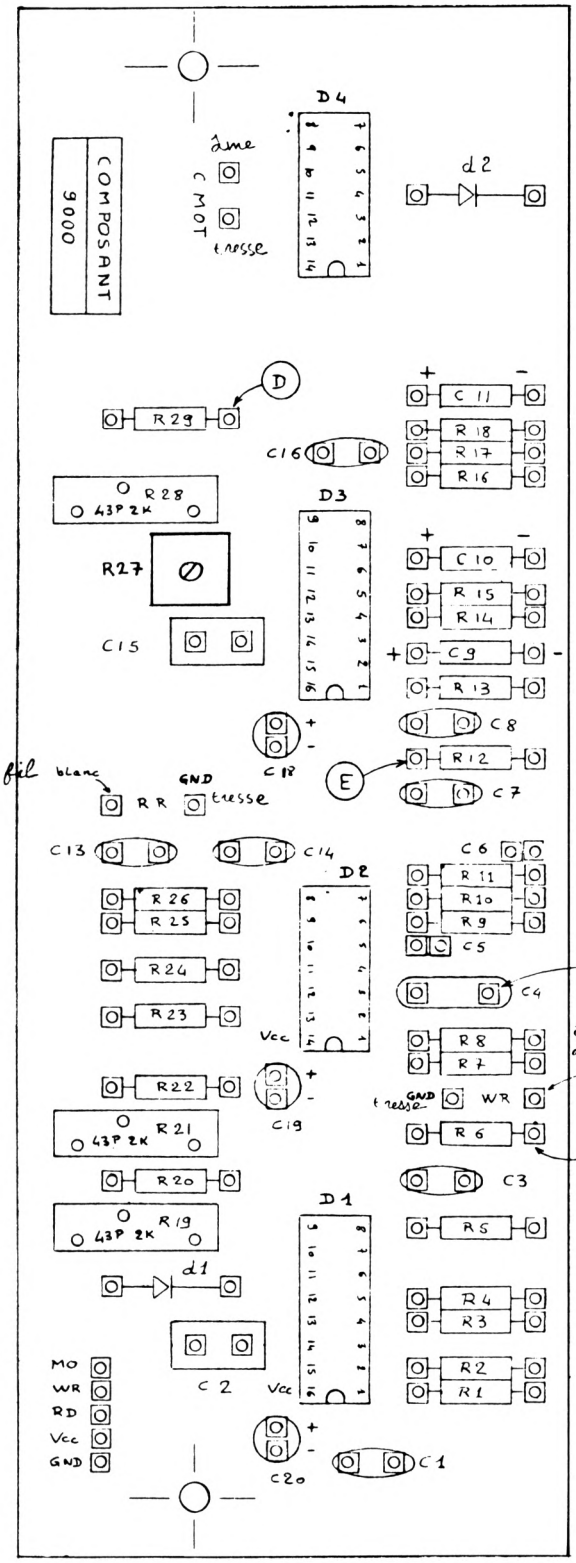


Vue de derrière (côté à souder)

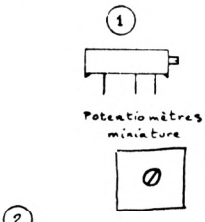
CARTES 4010 et 9000

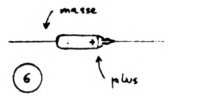


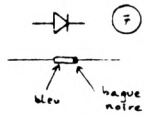
INTERFACE MAGNETOPHONE 9000



N° figure	Liste composants		
	D1 D3	XR 210	sur support
	D2	LM 3900	
	D4	DS1A05A	sur support
	R1	3,9 K	
	R2	10 K	
	R3	10 K	
	R4	10 K	
	R5	2 K	
	R6	220 K	
	R7	47 Ω	
	R8	1 K	
	R9	10 K	
	R10	5,6 K	
	R11	10 K	
	R12	3,9 K	
	R13	7,5 K	
	R14	3,9 K	
	R15	1 K	
	R16	3,9 K	
	R17	3,9 K	
	R18	1,5 K	
1	R19	2 K	
	R20	510 Ω	
1	R21	2 K	
	R22	220 Ω	
	R23	220 K	
	R24	22 K	
	R25	10 K	
	R26	22 K	
	R27	pot. 10K	
1	R28	2 K	
	R29	3,9 K	
2	C1	0,1 µf	
5	C2	68 nf	
2	C3	0,1 µf	
3	C4	0,47 µf	
4	C5	2,2 nf	
4	C6	10 nf	
2	C7	0,1 µf	
2	C8	0,1 µf	
6	C9	1 µf	
6	C10	1 µf	
6	C11	1 µf	
2	C13	0,1 µf	
2	C14	0,1 µf	
5	C15	68 nf	
2	C16	0,1 µf	
	C18, C19, C20	goutte tantale 2,2 µf	
7	d1	diode	
	d2		

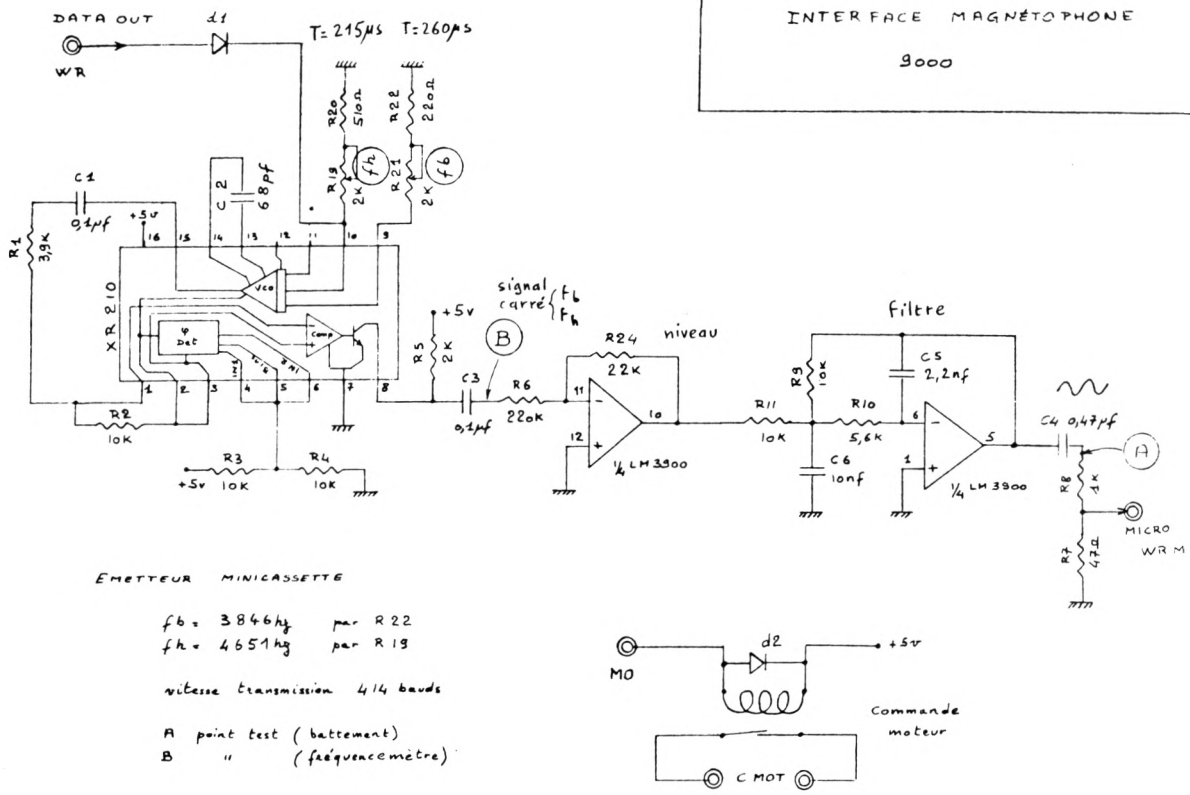






A B D E points tests

INTERFACE MAGNÉTOPHONE 9000

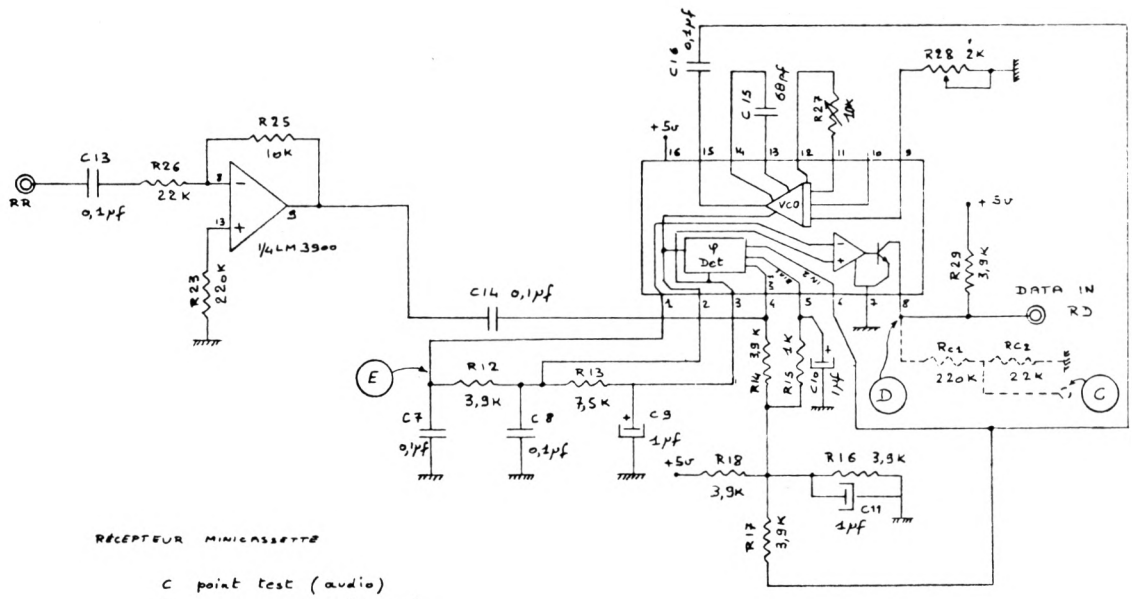


EMETTEUR MINICASSETTE

fb = 3846hz par R22
fh = 4651hz par R19

vitesse transmission 414 bauds

A point test (battement)
B " (fréquencemètre)



RÉCEPTEUR MINICASSETTE

C point test (audio)
D " (oscilloscope)
E " (")

```
STOP
END
```

ce qui paraît curieux au premier abord. Cette habitude de programmation permet d'arrêter un programme sur STOP, ce qui ne détruit pas les variables. On peut alors les examiner à loisir, rappelons qu'il suffit de faire

```
DISP variable [,variable...]
```

pour voir les valeurs des variables s'afficher sur l'écran.

De même, on peut réaffecter d'autres valeurs à des variables en faisant simplement par exemple

```
I = 27
```

Dès le RET, I prend sa nouvelle valeur, et la commande CONTINUE permet de reprendre l'exécution au début où l'on veut.

Reprenons notre description : on voit la commande

```
RUN
```

qui lance l'exécution du programme, lequel va donc écrire dix paires de variables sur la K7 après l'en-tête de dix secondes dû au MOVE, puis stopper le magnéto.

Ensuite, la commande

```
RECORD
```

va provoquer l'enregistrement du programme lui-même sur la bande, derrière les données, avec un inter-bloc assez long écrit automatiquement.

On entre ensuite un nouveau programme au clavier. Remarquons que

```
SEQUENCE 1Ø,5
```

est exactement équivalent à

```
SEQUENCE 1Ø
```

étant donné l'option par défaut 5 par 5. Les numéros de ligne étant exactement les mêmes que ceux de l'ancien programme, ce dernier va être complètement écrasé et remplacé par le nouveau.

La commande

```
MOVE 1Ø
```

est utilisée pour rembobiner manuellement la K7 comme vu, puis on tape la commande

```
RUN
```

Le programme s'exécute alors, et lit ce qui était écrit en début de K7, c'est-à-dire les dix paires de données. On remarque que la liste de variables du LOAD DATA est identique *en structure* à celle du RECORD DATA, mais avec des noms de variables ici différents. L'instruction

```
DISP X,Y
```

affiche les variables sur la TV, et à cause du PRINT ON nous les voyons apparaître sur le listage. Le magnéto est arrêté par le programme dès que toutes les variables sont lues. Nous trouvons ensuite

```
LOAD
```

Cette commande lit le programme qui se trouve en position sur la K7 : c'est notre ancien programme. Dès que le signe > est revenu, nous donnons la commande

```
LISTP
```

qui nous fait un listage de ce que nous avons maintenant en mémoire centrale : l'ancien programme qui a complètement écrasé le nouveau. Enfin

```
MOVE 1Ø
```

nous laisse à nouveau le temps de rembobiner... avant que > apparaisse.

Voilà finie notre petite gymnastique d'assouplissement... cassette ! Nous espérons qu'elle aura illustré simplement les possibilités fondamentales de cet interface peu coûteux mais déjà puissant.

7 - 8. LES MICRODISQUETTES

Avec ces unités périphériques, nous abordons un seuil supérieur de performances pour le système Alcyane, en raison :

— de la rapidité du débit d'information : un programme dont le chargement demande trois minutes et demie avec une K7 est maintenant chargé en trois secondes,
— de l'accès sélectif : au lieu de défiler obligatoirement les fichiers en entier, on peut accéder directement à un enregistrement,
— de la sécurité de fonctionnement : si bon soit le MK7, une K7 reste une K7, c'est-à-dire un objet assez mécanique et délicat. Le disque souple est beaucoup plus simple, et sa fiabilité est élevée.

7 - 8 - 1. Historique du disque souple

Rappelons que c'est IBM qui est à l'origine du disque souple, utilisé initialement comme support à haute fiabilité pour le chargement des microprogrammes des unités centrales 370. Ces der-

ALCYANE / BASIC VERSION 10

```
SEQUENCE 10
0010 MOVE 10
0015 FOR I=1 TO 10
0020 A=I:B=I*I
0025 RECORD DATA A,B
0030 NEXT I
0035 STOP
0040 END
```

programme entré au clavier: il écrit des résultats sur la K7

```
RUN
```

exécution

```
RECORD
```

enregistrement

```
SEQUENCE 10,5
0010 FOR I=1 TO 10
0015 LOAD DATA X,Y
0020 DISP X,Y
0025 NEXT I
0030 STOP
0035 END
```

également entré au clavier: il lit des données sur la K7

```
MOVE 10
```

rembobinage manuel

```
RUN
```

exécution

```
1                 1
2                 4
3                 9
4                 16
5                 25
6                 36
7                 49
8                 64
9                 81
10                100
```

```
LOAD
```

chargement du premier programme à partir de la K7

```
LISTP
0010 MOVE 10
0015 FOR I=1 TO 10
0020 A=I : B=I*I
0025 RECORD DATA A,B
0030 NEXT I
0035 STOP
0040 END
```

listage

```
MOVE 10
```

rembobinage manuel

nières ne contiennent qu'un bootstrap minimum destiné à lire le disque souple sur lequel sont stockés les microprogrammes matérialisant les instructions de la machine, et certaines fonctions fondamentales ou de diagnostic. Le contenu du disque est donc chargé dans la mémoire de commande, distincte de la mémoire centrale : le 370 est alors capable d'effectuer un IPL (initial programme load) pour charger le software.

La diskette (ou disquette, ou mini-disque, ou disque souple, ou floppy, équivalent du dernier terme) fut annoncée officiellement ensuite par IBM, le 22 janvier 1973, comme élément principal du système de saisie et de traitement de données 3740, avec le but clair de remplacer à terme la carte perforée.

Shugart, puis Memorex, Calcomp, etc. annoncèrent à leur tour des unités à disques souples. Une première évolution se fit vers le haut, Shugart (qui utilise des têtes de lecture/écriture vitro-céramiques personnelles) mettant au point la double densité : 600 000 octets par diskette, au prix d'une électronique de conception élaborée. Avec un bon décalage, d'autres constructeurs les uns après les autres rattrapèrent le mouvement.

Devenu leader du marché indépendant des diskettes, nanti d'une réputation justifiée de qualité, Shugart innova encore, mais cette fois avec une évolution vers un marché plus vaste : le 8 septembre 1976 fut annoncée la mini-diskette (que pour éviter des confusions nous appellerons plutôt micro-diskette) d'une capacité de 100 000 octets environ, réalisée dans le même esprit de qualité, mais nettement plus compacte comme unité et nettement moins coûteuse, à la suite d'astucieuses simplifications de mécanisme. Ces unités ont un temps moyen annoncé entre pannes de *8000 heures de fonctionnement !*

Une liaison directe MBC-Shugart, bien antérieure à cette date, nous permit d'étudier pour annonce au Sicob 1976, un sous-système complet hard et soft de micro-diskettes, ce périphérique nous étant apparu comme particulièrement adapté à Alcyane.

7 - 8 - 2. Le support

La micro-diskette se présente sous la forme d'un étui carré de 133 mm de côté contenant un disque souple de 130 millimètres de diamètre. Ce disque, en mylar épais de huit centièmes de millimètre recouvert d'une couche magnétisable à l'oxyde de fer, tourne toujours dans l'étui qui contient un lubrifiant spécial. Un trou central permet au moyeu de l'unité d'entraînement de positionner précisément le disque et de le faire tourner, à 300 tours/minute (donc 5/s) : dès son insertion dans l'unité, après fermeture, le disque est opérationnel, sa masse très faible lui permettant de se mettre en rotation sans délai appréciable. De la même façon, arrêt et extraction sont quasi immédiats.

Comme l'indique le croquis, la micro-diskette comprend 35 pistes, numérotées de 00 à 34, la piste 00 étant la piste extérieure. Une ouverture allongée dans l'étui permet à la tête de lecture/écriture d'accéder aux pistes. Près du centre, le disque est perforé de 16 trous également espacés repérant les débuts de secteurs, dont 16 secteurs par piste, 560 au total. Un 17e trou, placé entre deux trous de secteurs, le trou index, permet de repérer le premier secteur de chaque piste. La détection de ces trous est assurée par une LED et un photo-transistor, sans aucun contact. Toutes ces opérations sont sous contrôle du software.

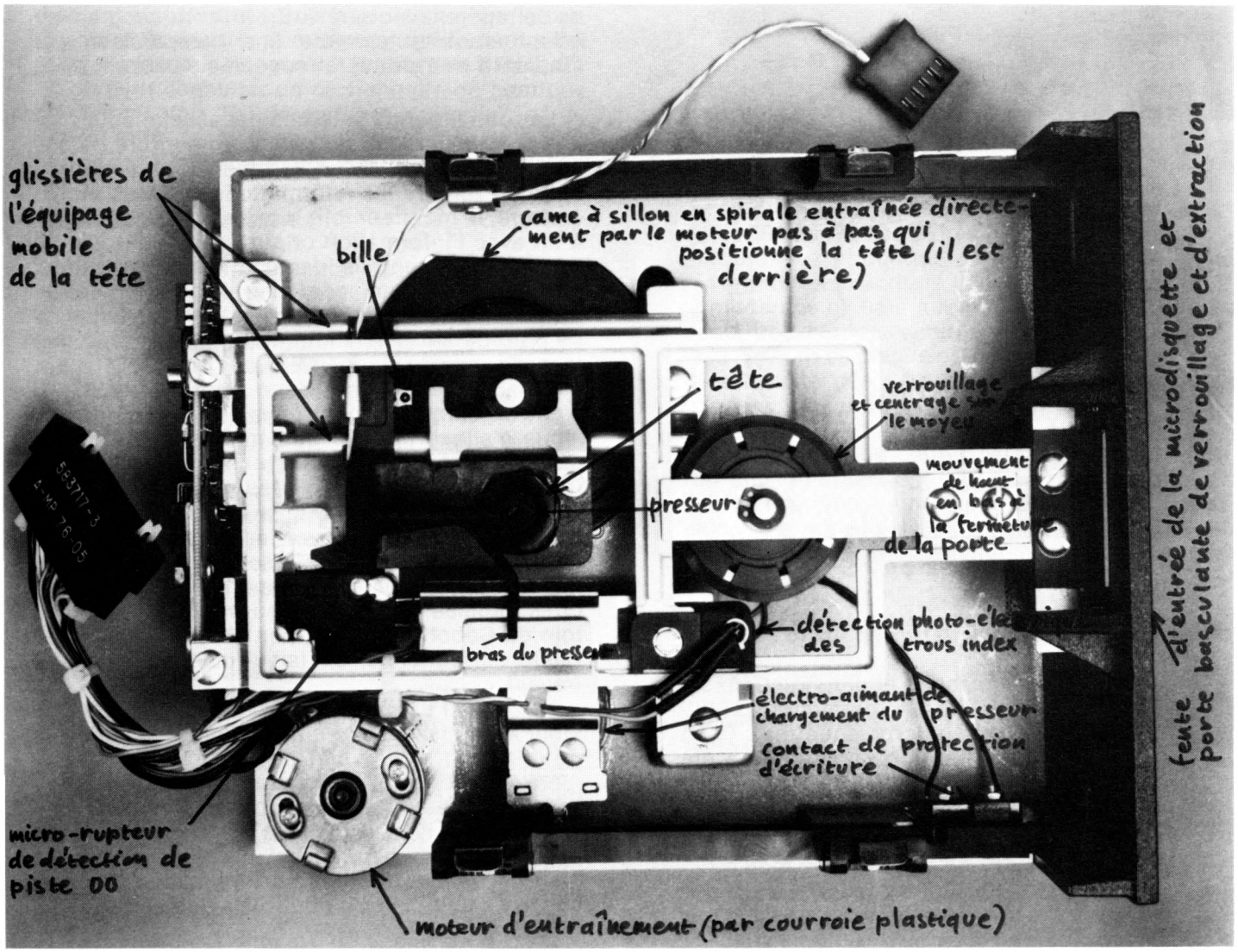
Chaque secteur est structuré de la façon suivante :

— adresse piste,
— adresse secteur,
— 168 octets (maximum) de données,
— adresses de chaînage et contrôle, gérés par le software.

La capacité nette de la micro-diskette est donc de 2688 octets par piste, soit 94 080 octets au total. L'insertion et l'extraction étant très rapides, un simple bac placé près de la machine et contenant quelques douzaines de disques permet de traiter des fichiers déjà importants stockés dans un faible volume.

A ce propos, rappelons les conseils élémentaires pour les fichiers magnétiques :

— stocker à l'abri de la poussière,
— ne pas écrire sur l'étui de la diskette avec un stylo à bille ou un crayon : la dureté de la pointe peut détériorer le disque. Utiliser seulement un crayon feutre, et mieux encore : marquer seulement un numéro sur l'étui, et le même sur l'éti-



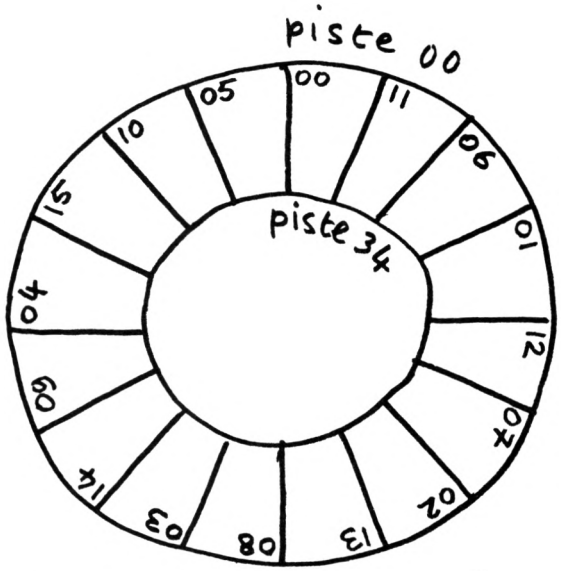
quette de la pochette dans laquelle il est livré, les libellés divers étant écrits seulement sur la pochette quand l'étui n'est pas dedans,
— stocker de façon à ce que l'étui ne se déforme pas,
— éviter le contact avec des objets magnétiques (aimants de fermeture de placard ou tiroirs, etc.), la forte chaleur...
— n'oublions pas que, contrairement aux disques rigides à têtes volantes, les diskettes s'usent (lentement) et sont donc malgré tout un support consommable.

Ceci dit, la disquette est un support d'une fiabilité remarquable. Notons tout d'abord que la tête polie de lecture/écriture ne touche la surface *que* pendant ces deux opérations.

La durée de vie d'une piste est (en théorie) de trois millions de passages en lecture ou en écriture, tête chargée par conséquent. Le taux d'erreurs fugaces (disparaissant en répétant l'opération de lecture ou d'écriture) est de même de un bit sur cent millions. Le taux d'erreurs persistantes est de un sur cent milliards... de très loin inférieur au taux d'erreurs existant dans les données ! Ces chiffres sont au bas mot cent fois meilleurs que pour les cassettes informatiques, que nous voyons condamnées à court terme : leurs unités sont plus chères, le support est plus cher, le temps d'accès, le débit et la fiabilité sont infiniment moins bons.

On a remarqué sur le croquis la disposition des secteurs : ils sont numérotés de 00 à 15, de trois en trois.

Cette disposition particulière accélère notablement les accès, en donnant le temps au programme de réaliser les opérations souhaitées entre l'arrivée sous la tête de lecture de deux secteurs de numéros consécutifs, ce qui permet de ne pas perdre un tour complet à chaque fois comme ce serait le cas avec une numérotation séquentielle.



Numérotation entrelacée des secteurs

Utilisation des microdisquettes en Basic

Une dizaine de commandes et d'instructions intégrées dans le Basic d'Alcyane permettent d'utiliser commodément les microdisquettes. Dans leurs définitions une attention particulière a été apportée aux questions de facilité de programmation, de taux de remplissage et de sécurité des fichiers.

7 - 8 - 3. Formattage et sécurité

Revenons sur ce point, qui a été seulement évoqué brièvement lorsque nous avons parlé du contenu des secteurs.

Chacun d'eux se présente donc de la façon suivante :

n°P	n°S	II	SYST	DONNÉES
I		//////	III	IV

La zone I représente l'étiquette du secteur : elle contient le numéro de la piste sur laquelle se trouve le secteur (00 à 34), suivi du numéro du secteur sur ladite piste (00 à 15).

La présence de ces étiquettes permet un excellent contrôle par le moniteur (le système, si l'on veut) intégré au Basic, pour vérifier si un déplacement commandé au moteur pas à pas de la tête nous a amenés sur la bonne piste (ce n'est pas que nous mettions en doute la qualité de la mécanique Shugart... mais nous tenons à cette réputation de "Rolls Royce du kit" que nos clients nous donnent !). Le numéro de secteur permet également de se repérer avec sûreté. Enfin, votre programme effectuera bien entendu une comparaison sur l'indicatif de l'article retrouvé, mais ceci est de la routine.

La zone II donne le temps nécessaire pour passer éventuellement de lecture en écriture : ainsi, après avoir lu et vérifié l'étiquette, on écrira ensuite en zone IV les informations désirées. Après écriture, le moniteur relit la zone et la compare bit à bit avec l'information originale qui est toujours en mémoire centrale dans le tampon d'entrée/sortie. En cas de discordance, l'opération est répétée jusqu'à cinq fois avant de signaler l'existence d'un défaut persistant (nous verrons plus tard la manœuvre de récupération dans ce cas).

La zone IV contient donc nos informations : c'est un enregistrement physique de 160 octets. Nous n'allons pas discuter maintenant en détail de sa composition : contentons-nous de dire que les enregistrements logiques (les articles) peuvent être plus courts ou plus longs que l'enregistrement physique, et que l'écriture des diverses rubriques des articles peut se faire en numérique condensé, étendu alphanumérique ASCII, rubriques signées et virgulées ou non... on n'est pas loin du Cobol direz-vous, peut-être, mais c'est un Basic perfectionné, en fait étudié pour pouvoir *travailler.*

La zone III est une zone "système", non accessible à l'utilisateur, et qui contient des adresses de chaînage et une clef de contrôle : cette dernière est générée automatiquement avant toute opération d'écriture, et est écrite en même temps que la zone IV. On dispose donc d'un contrôle supplémentaire par rapport à celui, décrit plus haut, de relecture après écriture avec comparaison (il y a un troisième contrôle, nous en parlerons plus loin).

Lorsqu'on reçoit une microdisquette, elle est vierge. A nous donc d'écrire les étiquettes. Cette opération est appelée "formattage", et — rassurez-vous — s'effectue automatique par la commande Basic

```
FMT #
```

Ceci permet une appréciable économie à l'utilisateur qui n'est pas obligé d'acheter des disquettes formattées coûteuses au constructeur.

Mais avant d'aller plus loin, il convient maintenant de passer au problème d'organisations de fichiers.

7 - 8 - 4. Organisation de fichiers

Deux organisations très différents de fichiers peuvent être réalisées par les ordres Basic d'Alcyane : leur description esquissera leurs domaines d'application, mais il faut tenir compte qu'en programmation de nombreuses méthodes d'accès

peuvent s'appliquer à une organisation de fichier. Ces deux notions sont malheureusement souvent plus ou moins mélangées... et pourtant il est clair qu'on peut consulter un dictionnaire de bien des manières ! Voir "*L'Informatique Nouvelle*", n° de janvier et février 1973 pour une tentative de clarification du sujet dans la rubrique "Encyclopédie".

1. Fichiers non structurés

Les articles d'un tel fichier peuvent être de longueur absolument quelconque (de un octet à des dizaines de milliers d'octets, seule limite la place en mémoire centrale pour lire un article). Ils se rangent séquentiellement dans le fichier au fur et à mesure qu'on les introduit. Leur longueur peut être variable d'un article à l'autre. Rien n'est à spécifier au point de vue adresse sur disque, etc., le système se charge de tout. A l'enregistrement, chaque article sera automatiquement stocké immédiatement derrière le précédent, jointif, sans perte de place sur le disque. On ne peut bien entendu accéder aux fichiers que séquentiellement. Cette organisation sera entre autres utilisée pour le stockage des programmes.

2. Fichiers structurés

Ici encore, les articles peuvent être de longueur quelconque, mais identique pour tous, et spécifiée à l'avance en secteurs et/ou fractions imposées de secteurs. Ces contraintes permettent d'améliorer le temps de recherche, de multiplier les méthodes d'accès possibles, et de trier les fichiers par des programmes utilitaires (fournis) en assembleur : tout ceci bien sûr puisque le système peut, avec cette organisation, calculer la position exacte du $n_{ième}$ article d'un fichier sur le disque.

La piste zéro est utilisée par le système pour stocker les paramètres disque/fichiers. Ceci posé, voici les commandes et instructions correspondantes.

a) FMT #[(u [,dddd])]

Cette commande va formatter un disque vierge. La première partie optionnelle indique le numéro de l'unité de disque. La deuxième partie optionnelle indique le numéro que l'on veut attribuer au disque, en quatre chiffres. Ce numéro servira d'identification et permettra plus tard de vérifier par programme qu'on a introduit le bon disque dans la bonne unité.

Pour formatter un disque placé sur l'unité 2 en lui donnant le numéro 7496, on tapera donc

```
FMT #(2,7496)
```

Par défaut, la commande

```
FMT #
```

tout court formattera sur l'unité 1 un disque de numéro zéro.

La succession des opérations effectuées par cette commande est la suivante :

1) les 560 étiquettes de secteurs sont écrites,
2) toutes les étiquettes sont ensuite relues pour vérification d'écriture : on les voit défiler en même temps en haut à droite de l'écran TV (quatre chiffres décimaux, numéro de piste en tête),
3) le numéro de disque est écrit sur la piste zéro,
4) la totalité des secteurs est écrite puis relue pour vérification du bon état de la disquette.

La bonne fin de cette opération, qui dure environ cinq minutes, est signalée par le retour sur l'écran du signe du moniteur. Une dizaine de messages d'erreurs différents peuvent apparaître en cas de défauts, permettant un contrôle complet de l'opération.

b) OPEN #[(u [,dddd])] "ffffff"

Cette instruction permet de créer un fichier non structuré. Les deux parties optionnelles correspondent à celles du FMT, et nous les retrouverons dans d'autres instructions. Le numéro d'unité et le numéro de disque *peuvent être des variables,* ce qui ouvre d'intéressantes possibilités que nous verrons plus loin, *ou même des expressions,* de valeur maximale 65535.

Le nom de fichier comporte de un à six caractères alphanumériques, le double quote (guillement, double apostrophe) n'étant pas permis, puisqu'il sert de délimiteur de chaîne.

A l'exécution de cette instruction, le numéro de disque est tout d'abord contrôlé. Puis les paramètres du fichier sont stockés sur la piste zéro. Ensuite, à chaque écriture d'un article du fichier, les adresses d'implantation seront créées automatiquement.

Rappelons qu'une instruction peut être utilisée comme commande (l'inverse n'est pas vrai), et que par conséquent avant de rédiger le programme 1 de paye on exécutera sans numéro de ligne par exemple

```
OPEN #(2, 1004) "P1PAYE"
```

qui ouvrira sur le disque 1004 installé sur le drive 2 le module de la chaîne de programmes de paye intitulé P1PAYE.

c) OPEN #[(u [,dddd])] "ffffff" (a,b)

Cette instruction permet de créer un fichier structuré, et son début est identique à l'instruc-

tion précédente. Entre les parenthèses finales, on trouve deux zones séparées par une virgule :

— la première indique le nombre d'articles du fichier sur la disquette (le fichier peut s'étendre sur plusieurs disquettes),
— la seconde indique la taille de l'article (par multiples de 32 octets, voir exemples).

De même que pour l'OPEN déjà vu, numéro d'unité et numéro de disque peuvent être des variables. On peut écrire par exemple

```
OPEN #(2,F4) "STKPA" (680,4)
```

qui réservera entièrement un disque : 34 pistes par 16 secteurs avec des articles de 128 octets précédemment monté sur l'unité 2, le numéro de disque étant calculé par le programme et rangé dans la variable F4.

Dès qu'un OPEN est exécuté, la place nécessaire au fichier est physiquement réservée sur la disquette.

d) CATAL #[(u)]

Cette commande lit la piste zéro du disque, et affiche le numéro du disque suivi du tableau nominatif des fichiers avec leurs paramètres comme suit :

nom/nombre d'articles/long. art./1re piste/1er secteur

La troisième rubrique est systématiquement mise à zéro pour chaque fichier non structuré, ce qui permet de les reconnaître comme tels. CATAL permet donc de savoir ce que contient une disquette quelconque dès qu'on l'a mise dans l'unité : fameux pour les désordonnés ! Si l'on est en mode PRINT ON le catalogue s'imprimera aussi : bonne sécurité pour reconstituer un catalogue détruit par une coupure de courant arrivant (1 chance sur 100, mais elle existe) pendant sa mise à jour !

e) RECORD #[(u [,dddd])] "ffffff"

Cette instruction sert à écrire sur la disquette un programme qui est en mémoire. Sa syntaxe est identique à celle de l'OPEN non structuré. L'écriture la plus simple de l'instruction est

```
RECORD "PROG"
```

sous cette forme, elle écrit le programme en mémoire, qui a reçu le nom de PROG (déclaré au moment de l'OPEN correspondant), sur l'unité numéro un, *sans vérification du numéro de la disquette* qui est montée. Sous la forme

```
RECORD (2) "PROG"
```

elle l'écrira sur l'unité numéro deux, et sous la forme

```
RECORD (2,1010) "PROG"
```

le numéro de disquette sera contrôlé. Il peut aller de 1 à 65535.

f) LOAD #[(n° unité [,n° disquette])] "nom"

Voici l'instruction symétrique de la précédente : elle lit un programme sur la disquette et le charge en mémoire. Le programme qui était précédemment en mémoire (s'il y en avait un) est détruit, mais les valeurs de ses variables sont conservées.

Si le LOAD est effectué sans numéro de ligne (comme une commande), il faut ensuite faire un RUN pour démarrer le programme. S'il est sous forme d'instruction faisant partie d'un programme appelant, l'exécution du nouveau programme (le programme appelé) démarrera automatiquement dès le chargement terminé.

Une autre forme du LOAD permet de réaliser cet enchaînement automatique avec ré-initialisation complète des variables.

g) RECORD DATA #[(n° unité [,n° disquette])] "nom" (n) liste

Voici l'instruction destinée à l'écriture d'un enregistrement de données sur un fichier déjà ouvert par un OPEN structuré. Toute la première partie est familière, par contre la fin comporte un supplément de paramètres :

— n est le numéro d'ordre de l'enregistrement dans le fichier (ce sera le nième du fichier). Sa valeur numérique peut donc varier de 1 à a tel que précisé dans l'OPEN,
— la liste est une liste de variables quelconques. On pourra donc mettre par exemple :

```
110 N = 3
110 RECORD DATA #"CLIENT" (N) A$,B,X,B$,T()
```

ce qui nous écrira le troisième enregistrement du fichier CLIENT.

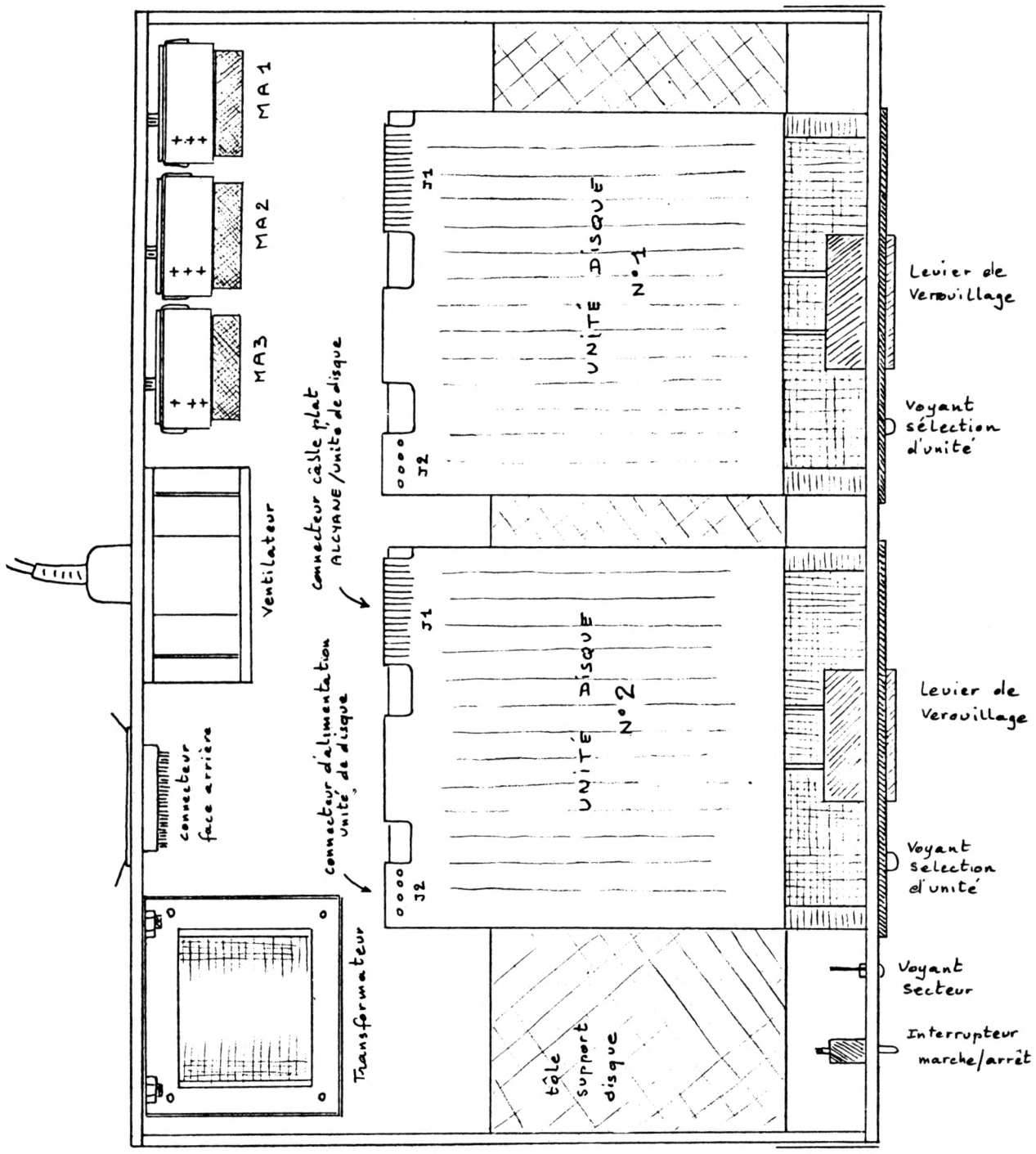
```
LOAD DATA #[(n° unité [,n° disquette])] "nom" (n) liste
```

C'est l'ordre symétrique qui permet de lire un enregistrement du fichier. Par exemple :

```
230 LOAD DATA #"CLIENT" (4) A$,B,X,B$,T()
```

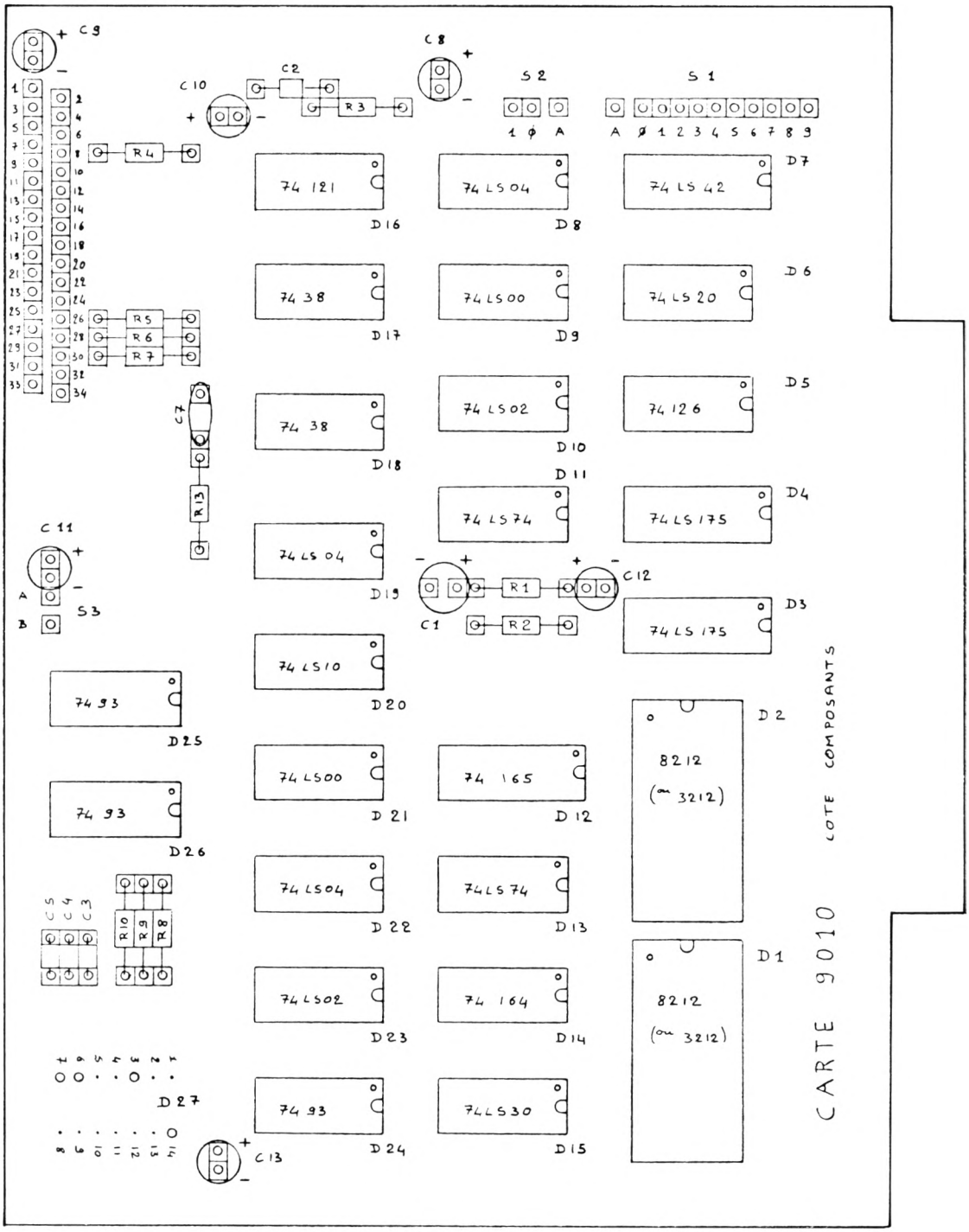
lira le quatrième enregistrement du fichier précédent.

Nous avons dit plus haut que le numéro d'unité et le numéro de disquette pouvaient être des variables (en fait, on peut même mettre des expressions !) ; il en est de même du nom du fichier, qui



IMPLANTATION GÉNÉRALE

MODULE MICRO DISQUETTE



R 1	10 KΩ	R 7	120 Ω	C 1	10pf tantale	C 8	2,2 pf tantale goutte
R 2	120 Ω	R 8	330 Ω	C 2	1 nf	C 9	"
R 3	10 KΩ	R 9	330 Ω	C 3	1 nf	C 10	"
R 4	120 Ω	R 10	330 Ω	C 4	1 nf	C 11	"
R 5	120 Ω	R 11	NON UTILISÉE	C 5	1 nf	C 12	"
R 6	120 Ω	R 12	NON UTILISÉE	C 6	NON UTILISÉE	C 13	"
		R 13	330 Ω	C 7	100 nf		

Strap S1 UNITE BCD (de ϕ à 9) relier A au plot correspondant

STRAP S2 DIZAINE BCD (ϕ ou 1) relier A au plot correspondant

STRAP S3 relier A à B

doit bien sûr être une variable alphanumérique. Par rapport à la syntaxe normale, ceci nécessite de rajouter une virgule et de supprimer les guillemets comme suit :

au lieu de "NOM" (x)
on met A$ (x)

ce qui ouvre d'intéressantes possibilités, en mettant des sous-chaînes indicées par exemple, etc. De même

```
LOAD DATA #(1 + 1,12*13) "STOCK" (X + Y/Z)A$,B
```

est parfaitement valide. On en imagine les possibilités. Pour créer un fichier séquentiel par exemple, on progresse N + 1 automatiquement en rebouclant sur le LOAD, ou on calcule le numéro d'ordre en fonction d'une des rubriques de l'enregistrement, etc, etc.

Montage des micro-disquettes

L'ensemble micro-disquettes comprend :

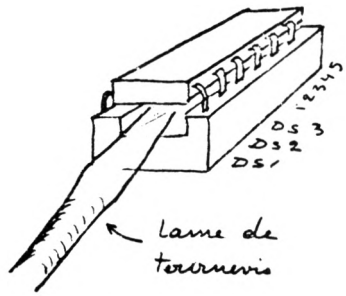
— une carte interface, qui prend sa place dans un des connecteurs de l'unité centrale Alcyane,
— un câble plat de liaison carte interface - carte disque,
— le coffret qui comprend lui-même :
  • la face arrière avec les alimentations,
  • une ou deux unités de disques,
— la mémoire REPROM 2708 de 1024 octets, à monter sur la carte processeur ; elle contient les routines de gestion physique des unités disquettes.

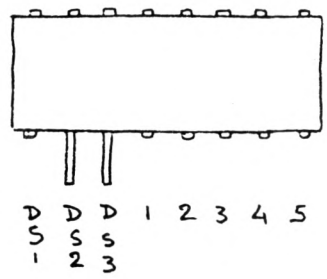
Ne jamais enlever la carte imprimée de son unité de disque. (Elle est préréglée en fonction de la tête lecture/écriture de chaque unité de disque).

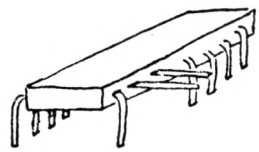
Numéro d'unité de disque

Un DIL spécial, enfiché dans un support, permet de spécifier la position physique de l'unité de disque, ainsi que son numéro, sur le bus de commande (câble plat de liaison avec l'interface). Enlevez-le à l'aide d'un petit tournevis pour effectuer les opérations suivantes :

— si vous possédez une seule unité de disque : mettre à l'horizontale les deux pattes correspondant à DS2 et DS3 (Drive select) et le réenficher,
— si vous possédez deux unités de disque : pour l'une mettre à l'horizontale les pattes correspondant à DS2, DS3 et le réenficher (cette unité est maintenant la n° 1, elle sera placée à droite dans le coffret) ; pour l'autre : mettre à l'horizontale les pattes correspondant à DS1, DS3, 1, 2, 3, 4, 5, et le réenficher. Cette unité est maintenant la n° 2 et est à mettre à gauche dans le coffret.







7 - 9. LES ENTREES/SORTIES...

7 - 9 - 1. Définitions et problèmes généraux

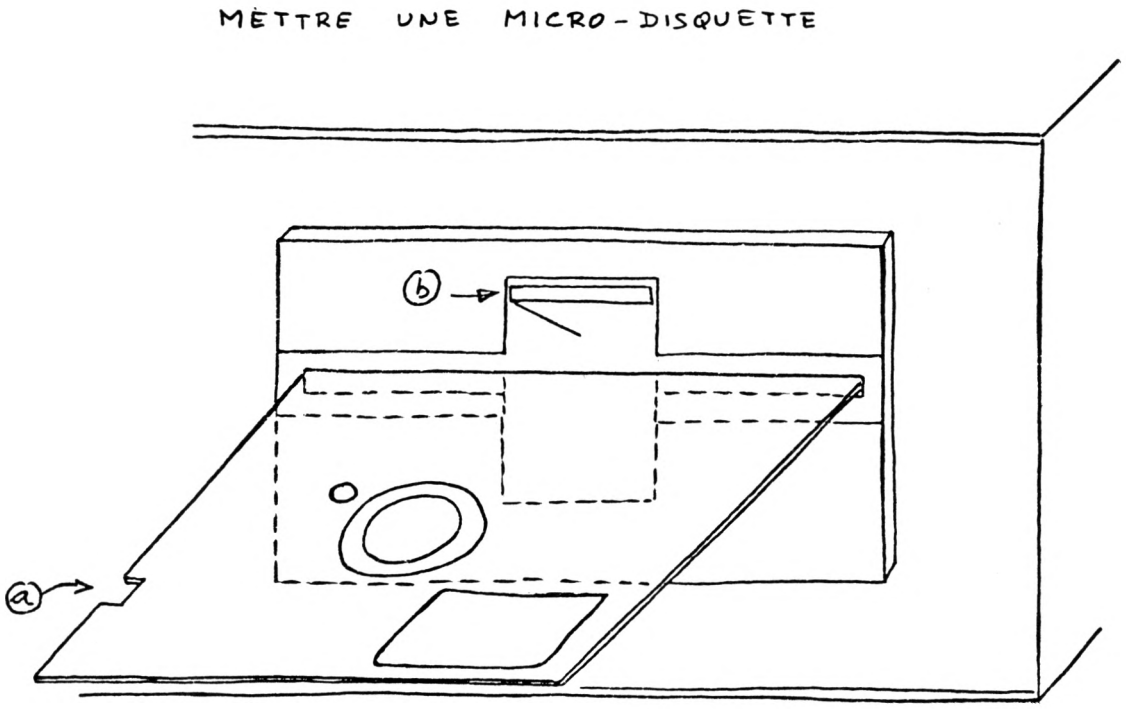
La coutume veut que l'on baptise ENTREES/SORTIES tous systèmes d'adaptation entre une UNITE CENTRALE d'ordinateur et ses PERIPHERIQUES.

Dans le même esprit, l'unité centrale est essentiellement constituée du module processeur et de ses modules mémoires.

Le module d'entrée/sortie est plus particulièrement appelé INTERFACE. Il possède deux grands rôles :

1 - Chaque périphérique, en fonction de ses particularités, utilise une méthode de dialogue précise de communication de l'information. Il est donc obligatoirement nécessaire d'adapter chacune de ces méthodes à la façon de procéder de l'unité centrale. C'est là un rôle de transcription de divers langages employés par chaque périphérique, à un seul langage, celui de l'unité centrale, rôle assuré par l'INTERFACE LOGIQUE.

2 - De par la diversité des périphériques utilisés au sein d'un même système de traitement de l'in-



formation, les signaux électriques, véhicule de l'information, n'utilisent pas toujours les mêmes conventions.

Par exemple, le 1 logique peut être représenté aussi bien par une tension de 5 volts, de 12 volts, ou bien de — 12 volts... ou même par la seule présence d'un courant !

Il est donc également nécessaire de compléter la partie interface logique par une adaptation de tension ou de puissance.

Cette partie reçoit le nom d'INTERFACE ELECTRIQUE.

Le Dilemme

Pour qu'un module d'entrée/sortie soit d'un emploi commode, il lui faut trois qualités antagonistes :

1 - une grande rapidité de transfert de l'information,

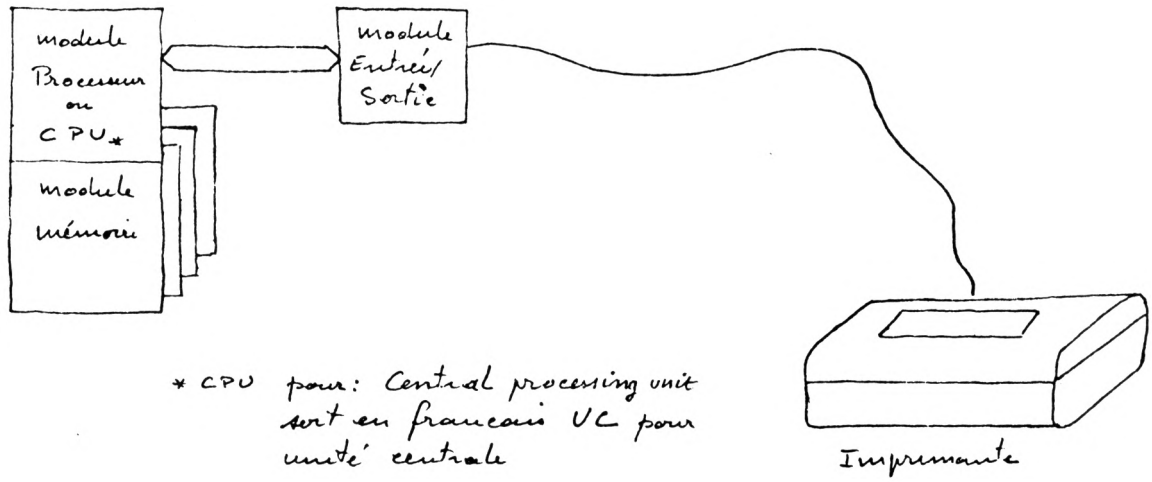
2 - l'emploi d'un câble de liaison non limité en longueur, afin de pouvoir installer le périphérique auquel il est lié selon notre désir.

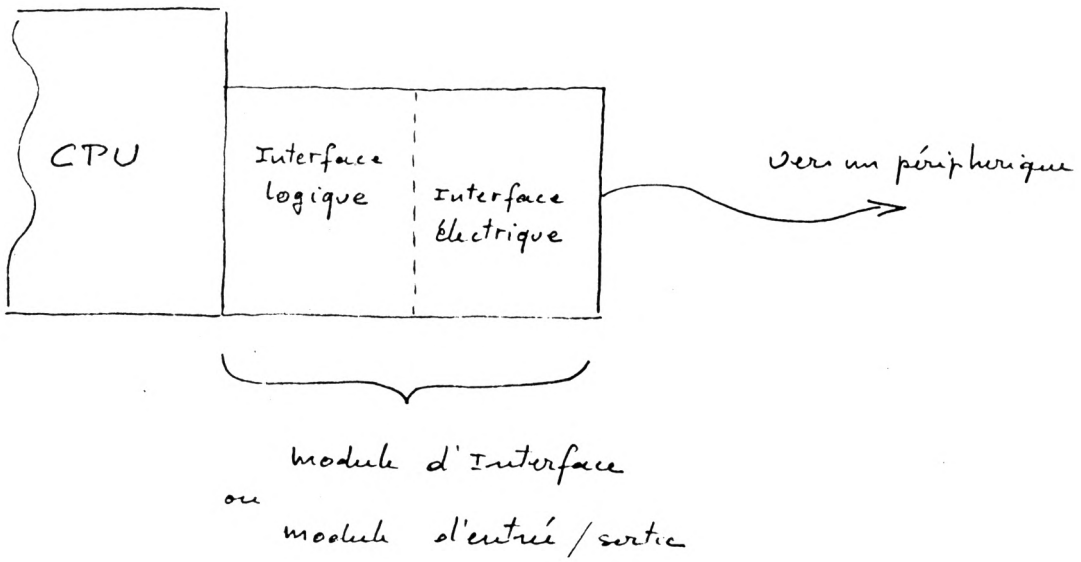
3 - le câble de liaison comprenant lui-même le moins de conducteurs possibles pour des raisons de coût (dont l'aspect est toujours négligé... sauf au moment de l'installation du système !).

En pratique, une grande rapidité de transfert n'est possible que sur quelques mètres de liaison avec des câbles comprenant un grand nombre de conducteurs (jusqu'à 40 dans certains cas).

Il est par contre assez évident que le transport de l'information sur plusieurs centaines de mètres, voire des kilomètres, n'est pas économiquement réalisable dans de telles conditions.

A eux seuls, ces deux critères distance/vitesse ont donné naissance à deux grandes familles d'interfaces.





Les interfaces dits parallèles, où chaque bloc d'information représenté par des mots (un octet ou 8 bits) transite d'un point à un autre en parallèle avec un conducteur par bit.

Les interfaces dits série, où chaque bit du mot transite l'un après l'autre sur un même fil.

La majorité des modules d'entrée/sortie d'Alcyane sont du type parallèle. Ils font l'objet d'études diverses dans la suite de ce texte, cas par cas.

Quelques principes

IN-OUT

Dans tout ce qui suit, nous adopterons la convention suivante :

le processeur est l'élément central par rapport auquel sont référencés les sens d'entrées/sorties.

C'est-à-dire :

tout ce qui rentre dans le processeur est une ENTREE (appelée IN en langue anglo-texane),

tout ce qui sort du processeur est une SORTIE (OUT).

C'est une règle impérative qui évite de nombreuses confusions !

ENTREE/SORTIE PARALLELE

Ce moyen de communication est réservé sur Alcyane à des distances courtes, *environ deux mètres maximum, trente mètres pour les liaisons CPU-CPU.*

ENTREE/SORTIE SERIE

Ce moyen est aussi bien utilisé pour des distances courtes que très longues. Il peut éventuellement s'adapter au réseau téléphonique privé ou

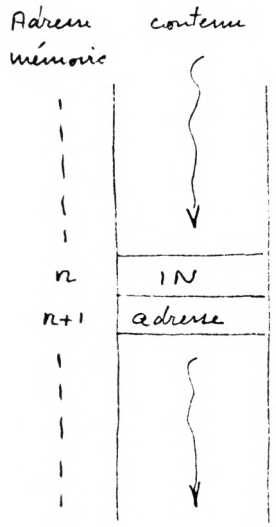
PTT, et donc s'affranchir ainsi complètement des problèmes de distance.

L'emploi des transmissions série par réseau téléphonique est soumis à des règles adoptées par presque tous les constructeurs d'équipement informatique. Elles sont regroupées sous le standard CCITT RS 232 C Avis V24... !

Les instructions d'entrée/sortie

Carte processeur 1001

Le lien entre Alcyane, unité centrale, et ses périphériques, est uniquement constitué par le registre A : l'Accumulateur. Son contenu est automatiquement transféré sur un module périphérique, ou, au contraire, chargé d'un module périphérique, lors d'une instruction OUT (sortie), ou IN (entrée).

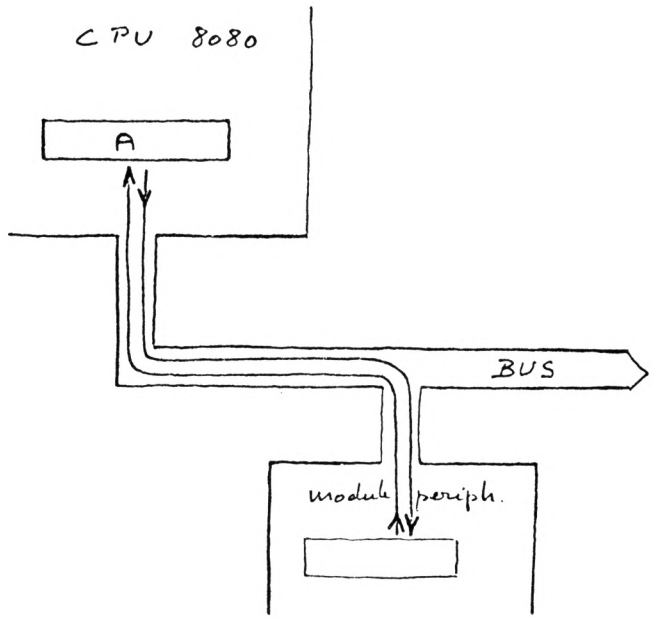


Ces deux instructions sont définies en mémoire par deux octets. Le premier spécifie l'opération à effecteur (IN ou OUT), le deuxième spécifie l'adresse du périphérique concerné.

Il en résulte que pour transférer une partie du contenu de la mémoire sur un périphérique (cas de l'imprimante par exemple), il est nécessaire de passer par l'accumulateur, puis vers le périphérique.

Adresse périphérique

Chaque périphérique est relié à la carte processeur d'Alcyane par un cheminement unique : le BUS. Mais chaque mot échangé sur ce bus est accompagné d'une adresse afin de définir le périphérique avec lequel la carte processeur est en train d'effectuer cet échange.

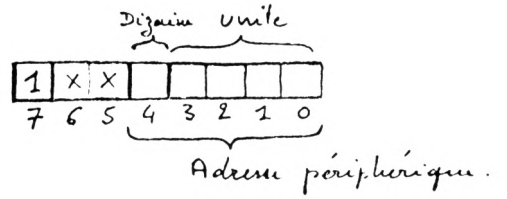


Une des fonctions importantes de la carte Module périphérique est le décodage de cette adresse : si elle reconnaît son adresse, elle doit se connecter au bus, et dans le cas contraire, se déconnecter du bus.

Structure de l'adresse d'entrée/sortie

L'adresse du périphérique peut être comprise entre 00 et 19, soit 20 périphériques maximum simultanément connectés sur Alcyane.

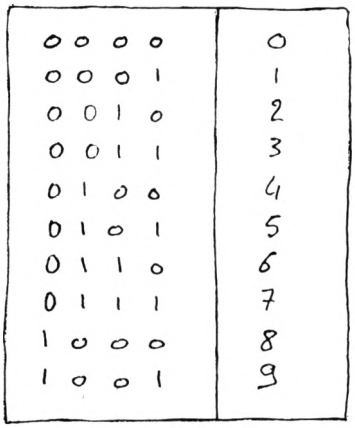
Le codage est fait en BCD (décimal codé binaire)
Le bit 7 doit toujours être à 1 (pour la carte CPU 1001).



Rôle des bits 6 et 5

Une carte module périphérique peut éventuellement effectuer plusieurs fonctions internes (exemple : la carte entrée/sortie universelle) ou être connectée elle-même à plusieurs périphériques (exemple : la carte combinée). Les bits 6 et 5 sont appelés adresse interne à la carte module périphérique, et spécifient les fonctions internes que la carte peut éventuellement posséder.

Le BCD



0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9

Les périphériques : notions de moniteur et de système

Les différents périphériques reliés à Alcyane constituent les moyens d'entrée/sortie de l'unité centrale, ou des moyens de stockage d'information au sens large (programmes ou données).

L'ensemble périphériques-unité centrale est appelé dans le jargon de l'informatique UN SYSTEME.

Pour que cet ensemble puisse être commodément utilisable, il faut lui adjoindre des programmes de gestion de chacun des périphériques (appelés routines de gestion, ou drivers en américain). La collection de ces routines de gestion telles que routine de gestion de l'imprimante, du disque souple, etc. constitue une partie essentielle du MONITEUR.

Si maintenant nous voulons utiliser un périphérique, l'imprimante par exemple, nous avons deux possibilités :

1. en l'absence de moniteur, nous écrirons un petit programme capable de mettre en route l'imprimante, puis de lui transmettre des données, et enfin soit de changer de ligne pour recommencer, soit de l'arrêter.

2. ou alors nous ferons appel au moniteur, et plus particulièrement à la routine de gestion de l'imprimante, par l'intermédiaire de commandes, c'est-à-dire de mots que le moniteur sait interpréter. Dans ce cas nous allons certainement utiliser un clavier pour introduire les commandes, ainsi qu'un écran de visualisation ou un téléimprimeur.

De tout ceci, il faut retenir que le software d'exploitation d'un système introduit une notion particulière, et sans relation avec le hardware, de spécialisation fonctionnelle des périphériques :

— WRITE
— READ

et ceci indifféremment des types de périphériques utilisés. Ils sont en quelque sorte "hors système".

Nous aurons l'occasion, au cours des pages suivantes, d'illustrer les points que nous venons de voir, qui résultent en fait des définitions formelles du langage de programmation.

Et pour bien illustrer ce dernier point, ajoutons qu'un périphérique "système" peut devenir "hors-système" dans certains cas, puisque, par exemple, l'imprimante peut être aussi bien actionnée par un PRINT que par un WRITE.

Structure générale de la machine

Un système Alcyane est constitué d'un certain nombre (qui commence à devenir important) de cartes ou modules.

A chaque module est attribuée une fonction bien particulière :
— module CPU
— module mémoire
— module périphérique pour imprimante
— module périphérique pour unité de disque
et bien d'autres...

Tous ces modules communiquent par une voie commune, le BUS, vers l'unique module, chef d'orchestre de l'ensemble, le CPU.

Ce chemin unique de l'information, donnée ou instruction, appelé BUS INFORMATION, est accompagné de lignes spécifiant d'une part la nature de l'information véhiculée, et d'autre part sa destination. Le premier groupe est appelé BUS COMMANDES, le deuxième BUS ADRESSE.

Les états logiques 1 ou 0 des lignes du bus de commande, pilotées par le module CPU, déterminent :
— le sens de transit de l'information sur le bus information, soit :
  • du CPU vers un autre module,
  • d'un module vers le CPU,
— spécifie si l'adresse présente sur le bus adresse est destinée aux modules mémoires, ou bien aux modules périphériques.

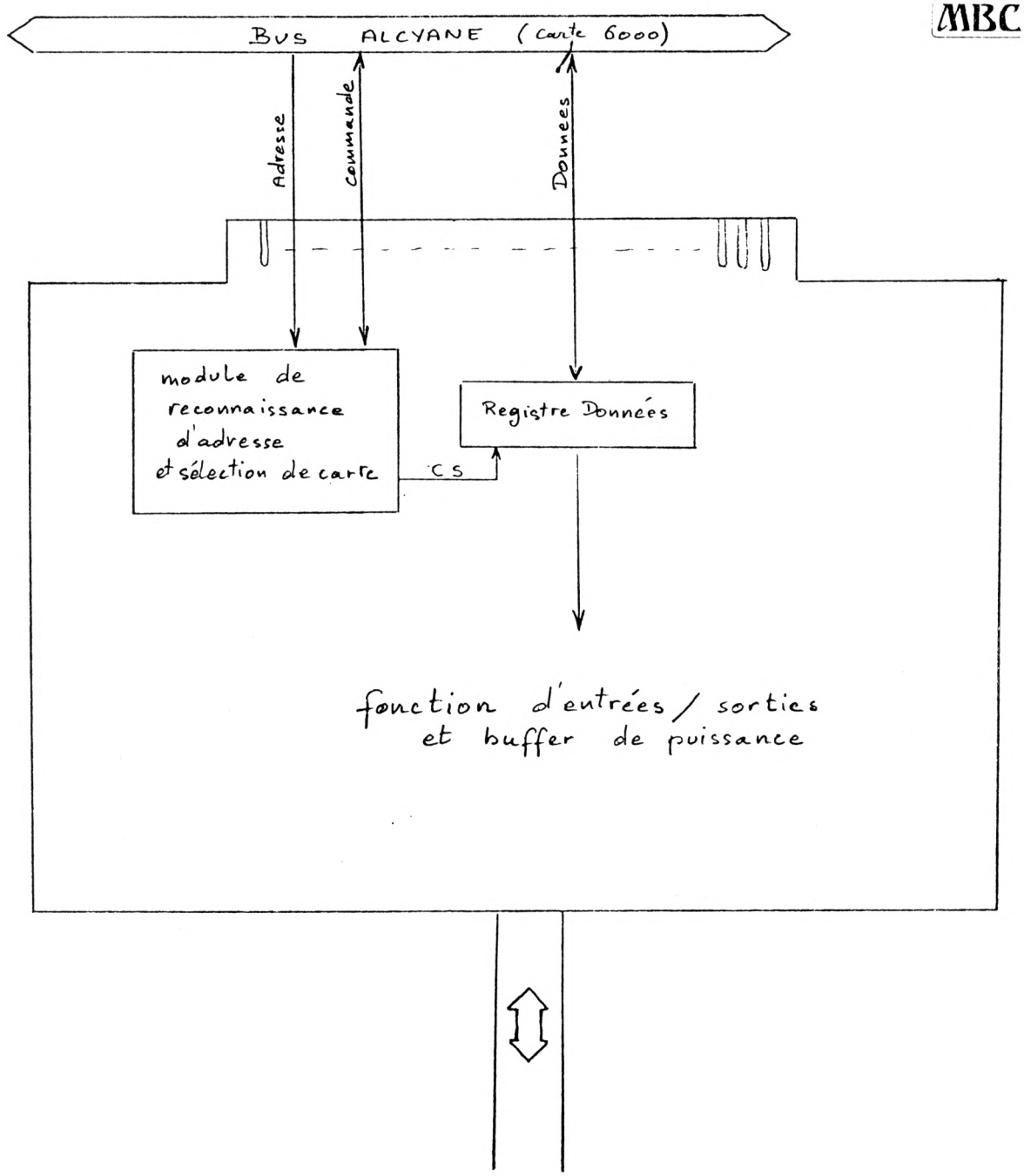
Le bus : signaux, adresses, considérations électriques

Les signaux d'entrée/sortie

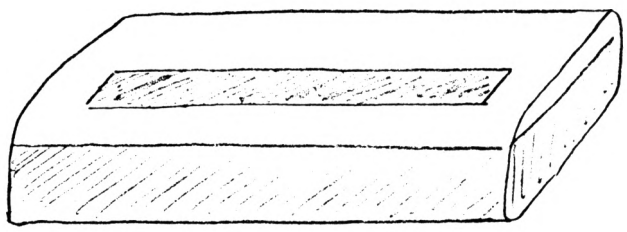
Le bus est utilisé aussi bien pour les accès mémoire, que les accès périphériques, par le processeur. La ligne $\overline{\text{MVA}}$ permet de différencier le type d'accès.

CARTE CPU 1001 : DESCRIPTION DES LIGNES

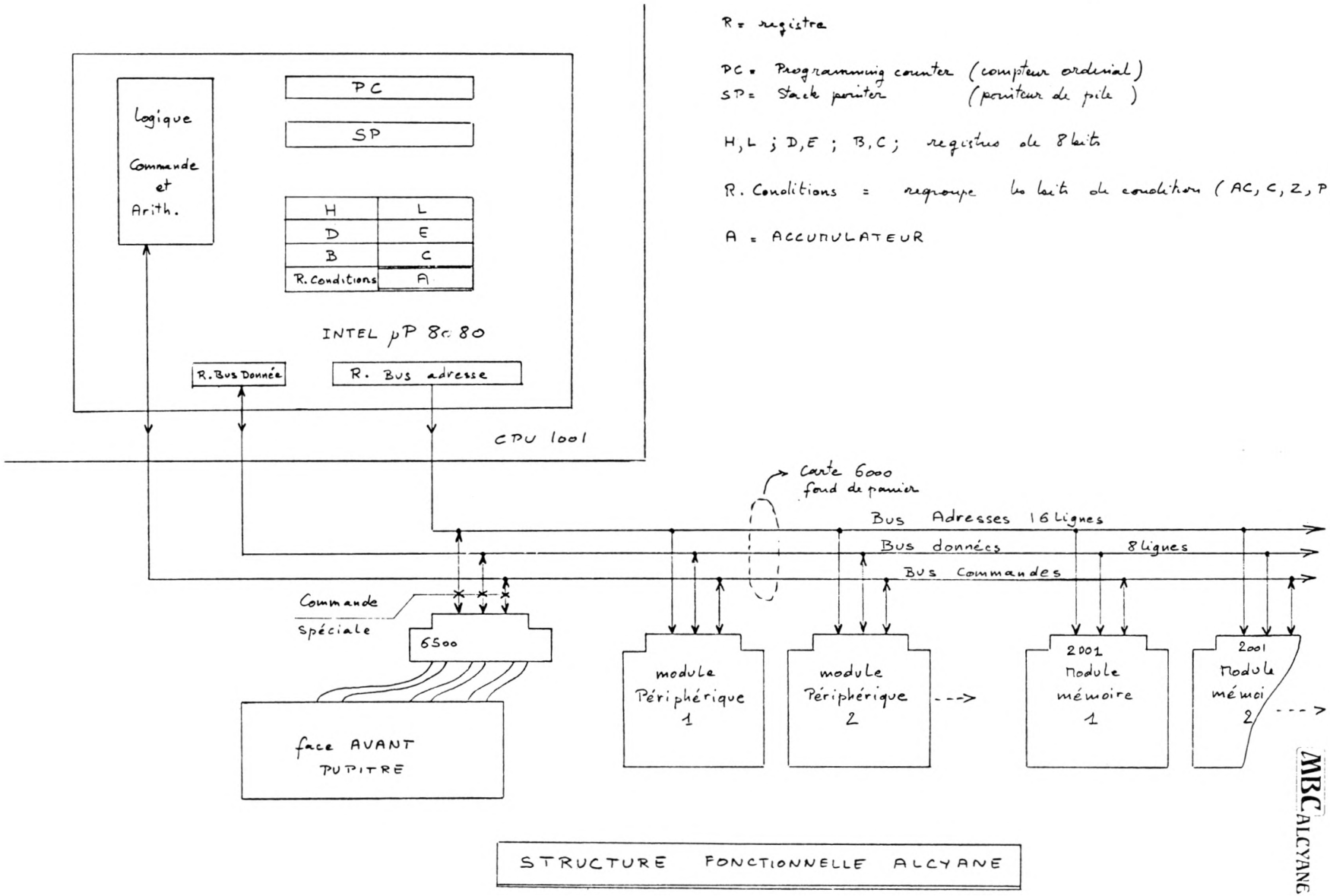
$\overline{\text{MVA}}$ (Memory Valid Address) actif bas
0 logique (0 volt) - accès mémoire
1 logique ($>$2 volts) - accès périphérique



PERIPHERIQUE



Imprimante



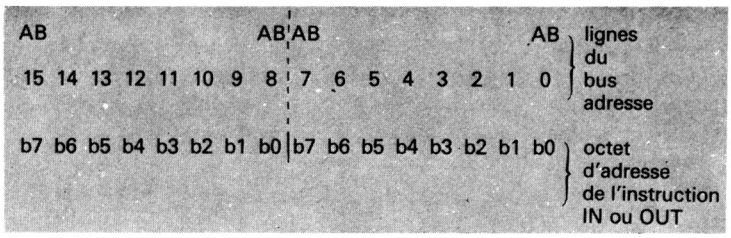
STRUCTURE FONCTIONNELLE ALCYANE

MBC ALCYANE

$\overline{I/OR}$ (Input Output Read) actif bas
Commande de lecture d'une carte module périphérique à l'adresse définie par AB0 - AB15

$\overline{I/OW}$ (Input Output Write) actif bas
Commande d'écriture sur une carte module périphérique à l'adresse définie par AB0 - AB15

AB0 - AB15 Lignes d'adresse du bus
Le contenu de l'octet d'adresse de l'instruction entrée/sortie est placé sur la partie basse *et* la partie haute des lignes adresse AB0 - AB15



Décodage de l'adresse périphérique

BCD Unité
AB0 à AB3 ou AB8 à AB11 sont décodés par un décodeur BCD du type 74 LS 42

BCD Dizaine
AB4 ou AB12 intervient par l'intermédiaire de deux inverseurs (le premier pour bufferiser la ligne) au niveau de la porte NAND 3 entrées.

MVA et AB7 (ou AB15)
Pour la carte processeur 1001, AB15 est directement appliqué sur $\overline{MVA}$ au moyen d'un strap sur la carte liaison.
Ceci limite le champ d'adressage de la carte processeur 1001 à 32 K octets (pour respecter la convention d'adressage mémoire-périphérique utilisée sur $\overline{MVA}$).

DB0 à DB7 (Data Bus) lignes de données

Ce sont des lignes trois états, elles possèdent la propriété d'être bi-directionnelles. Il faut donc faire très attention à ce que la carte périphérique ne soit pas elle-même en écriture sur le bus de données s'il est déjà sous contrôle d'un autre module. Si non, il y a conflit d'accès.

Pour éviter ce problème à coup sûr, il suffit de connecter le module périphérique (préalablement adressé) au bus données uniquement sur la commande $\overline{I/OR}$ (actif bas).

Au repos, c'est-à-dire le module non adressé, il peut être mis en position d'acceptation des données par le bus données.

Pour mieux voir ce problème, étudiez la partie CARTE COMBINEE plus haut.

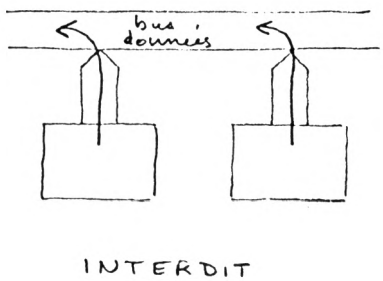
Considérations électriques

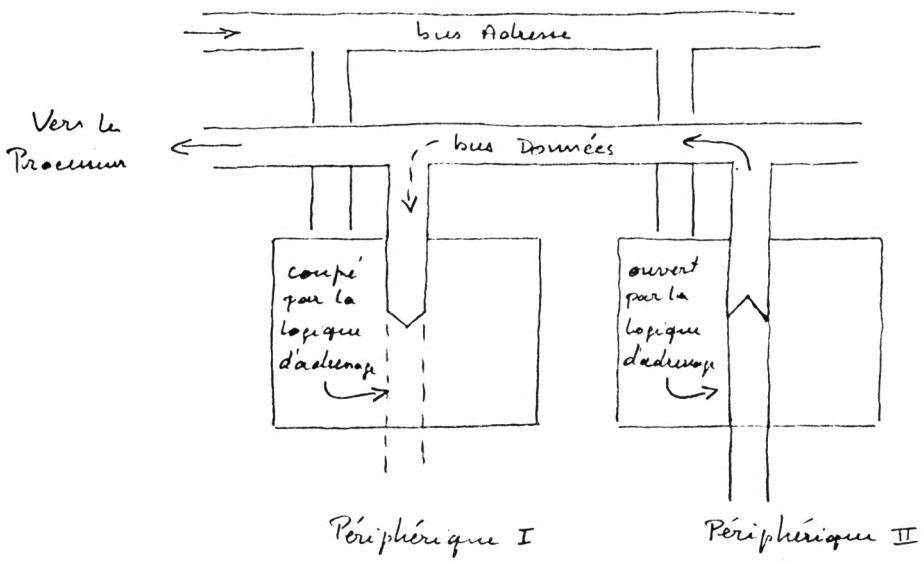
Toute ligne utilisée sur une carte périphérique, et provenant du Bus d'Alcyane, doit être impérativement bufférisée par un circuit TTL LS, ou ne voir qu'une seule charge LS.

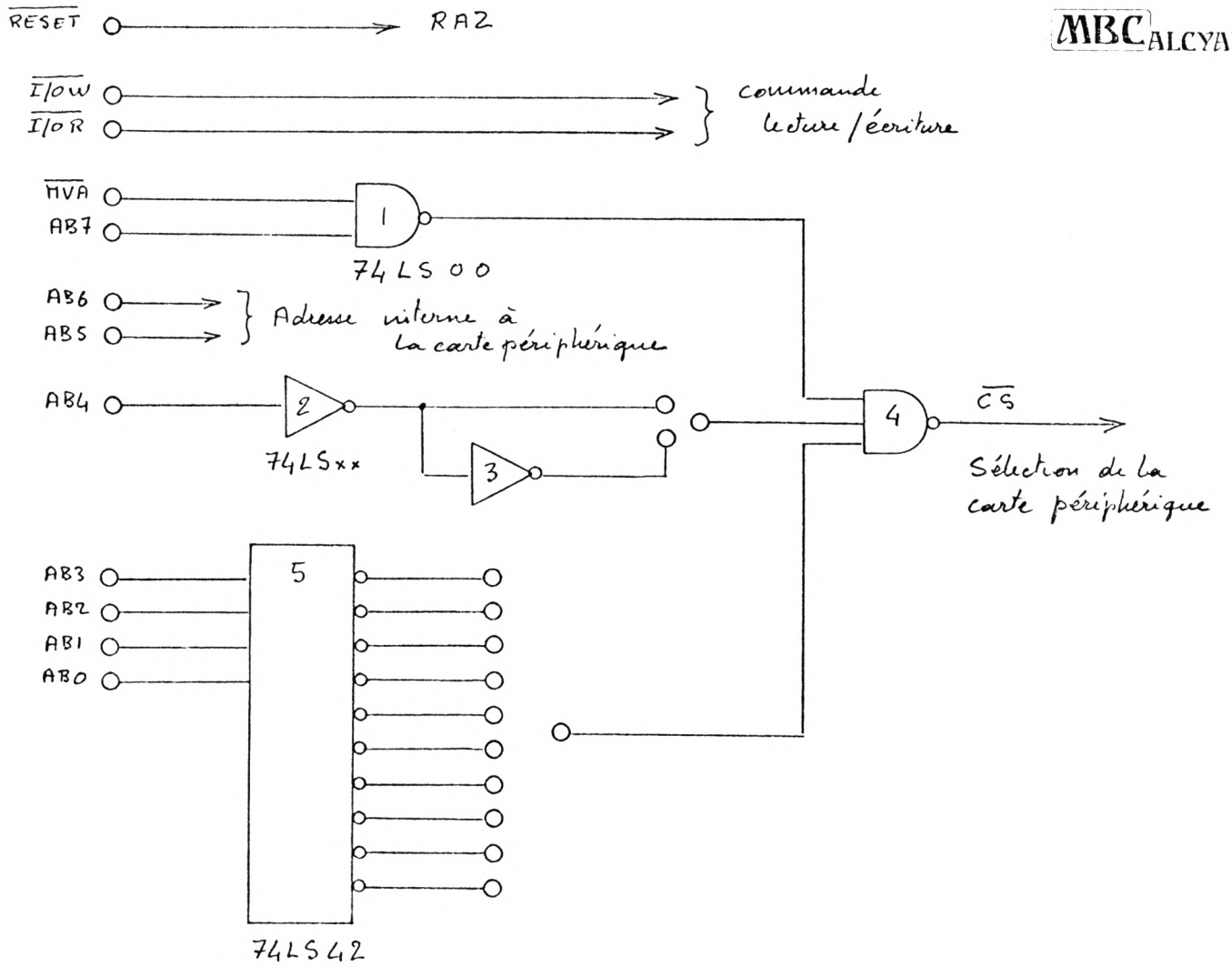
Le 8212

Ce circuit intégré comprend huit bascules type D bufferisées en trois états. C'est un des meilleurs circuits, en tant que caractéristiques électriques, pour bufferiser et mémoriser les lignes Adresses ou Données du bus.

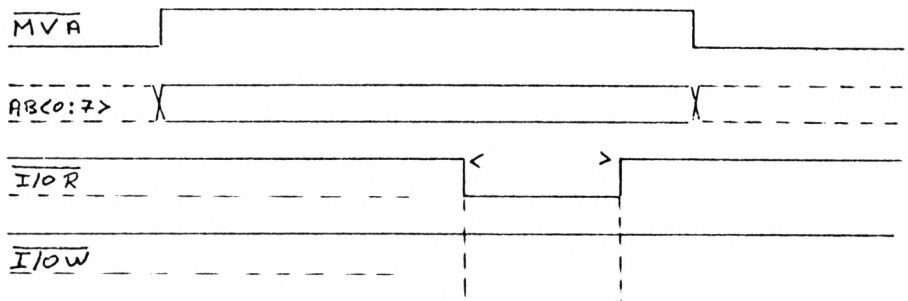
(Voir ses caractéristiques en annexe).



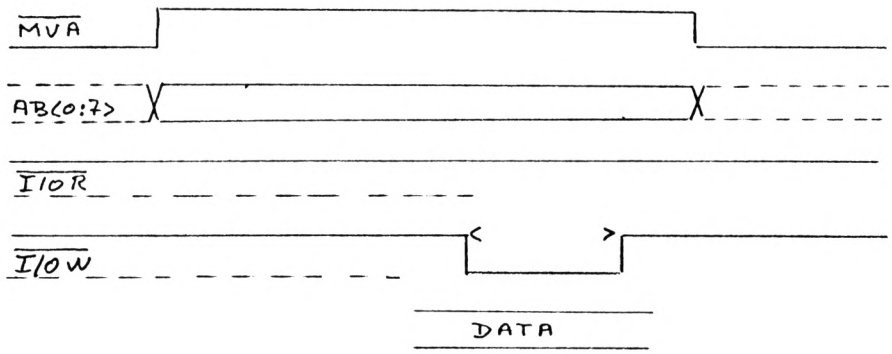




LECTURE



ÉCRITURE



7 - 9 - 2. Entrées/sorties parallèles

Module entrées/sorties universel

Structure générale du 8255

Ce module comprend trois registres généraux d'entrées/sorties : registres A, B, C dont les rôles sont non spécifiés à l'origine.

Ils peuvent donc être aussi bien des registres d'entrée, ou de sortie, et dans certains cas particuliers les deux à la fois (pour le registre A) et même voire certains de leurs bits utilisés comme commandes (pour le registre C).

En fonction du contenu du registre de commande, la logique de commande spécifie le mode d'emploi, et le rôle, de chacun des registres A, B, C.

Les bits 6 et 5 du mot adresse de l'instruction d'entrée ou de sortie (IN ou OUT) déterminent le registre avec lequel l'échange du contenu de l'accumulateur doit avoir lieu.

Soit : bit 6 5
- 0 0 Registre A
- 0 1 Registre B
- 1 0 Registre C
- 1 1 Registre d'état (ou registre du mot de commande).

avec cette restriction que le registre d'état ne peut être lu.

Les bits 0 à 4 du mot adresse des instructions d'entrées/sorties (IN ou OUT) donnent l'adresse à laquelle la carte module E/S a été physiquement placée par les straps S1 et S2.

Exemple :

```
MVI     A
FF
OUT
E1
```

chargement de la valeur FF (hexa) dans le registre de commande, la carte E/S étant à l'adresse périphérique 01 (ne pas oublier que le bit 7 du mot adresse E/S est toujours à 1).

Les modes de fonctionnement

Retour sur quelques principes fondamentaux !

Il n'existe que deux manières de dialoguer avec un périphérique (pour l'unité centrale, bien sûr).

1 - Considérer que le périphérique est toujours prêt et ne pas se préoccuper de son état (exemple : MK7 et clavier),

2 - ou au contraire, synchroniser les échanges de données sur la vitesse du périphérique (imprimante 80 colonnes...).

Il est évident que dans ce dernier cas, outre les lignes de données, des lignes spécialisées de cadencement du dialogue sont nécessaires.

Nous retrouverons, tout au long de l'explication des modes de fonctionnement de la carte E/S universelle, ce principe fondamental.

MODE 0

Est destiné à l'emploi des registres A, B, C à l'état brut, en considérant a priori que le périphérique est toujours prêt.

Chacun des registres peut être défini comme un registre d'entrée, ou de sortie (mais non les deux à la fois).

MODE 1

Dans ce mode, certains bits du registre C sont utilisés à cadencer le flux d'information entre les registres A et B, et un périphérique.

MODE 2

Fonctionnellement, il est identique au mode 1, mais permet en outre d'effectuer sur les mêmes huit bits du registre A, aussi bien des entrées que des sorties (fonctionnement bi-directionnel).

Spécifité des registres A, B, C (sur le plan fonctionnel)

REGISTRE C

C'est un registre assez curieusement traité,

— en premier, on peut le considérer comme l'ensemble de deux registres à 4 bits (mode 0),

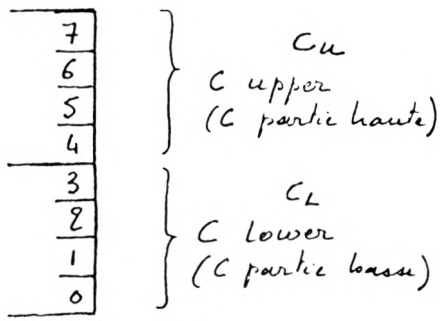
— en second, il est utilisé pour le contrôle du cadencement de l'échange d'information entre un périphérique et les registres A et B (modes 1, 2).

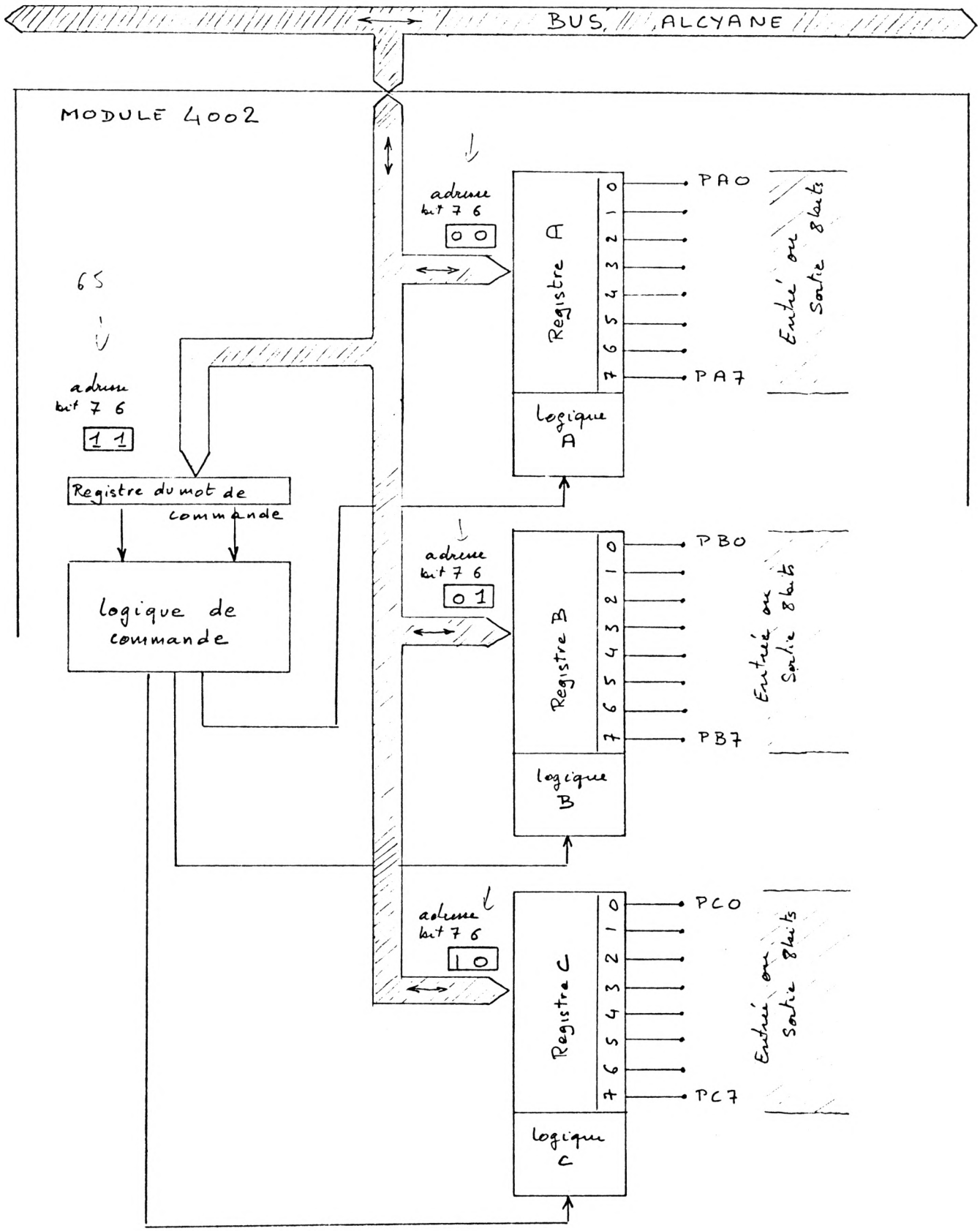
REGISTRE A

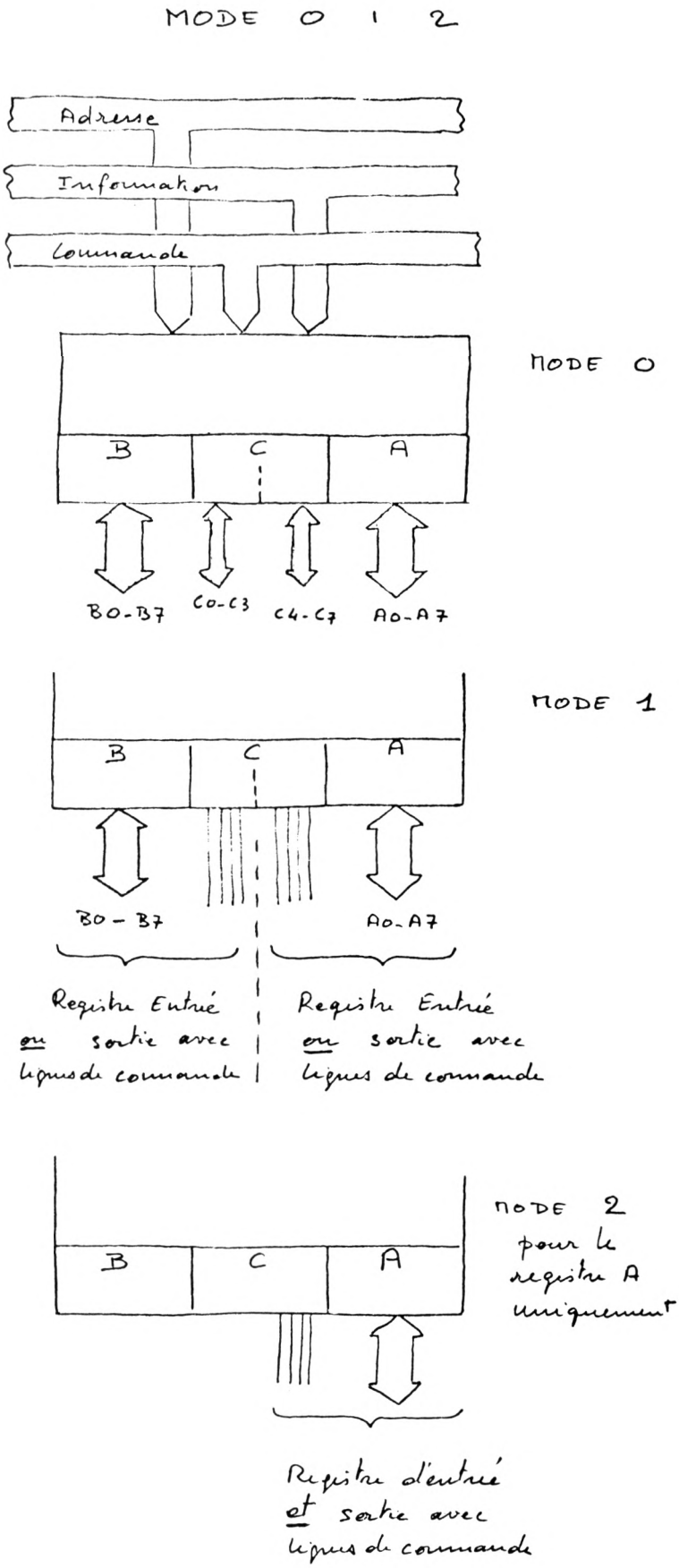
Il peut être utilisé en bi-directionnel (Mode 2).

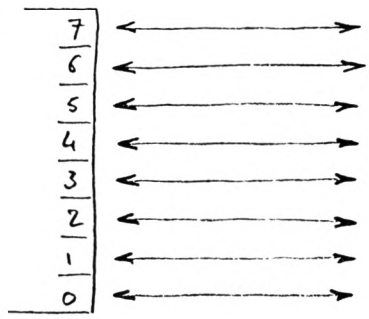
REGISTRE B

Rien de particulier.




BUS ALCYANE
MODULE 4002
adresse bit 7 6
0 0
Registre A
PA0
PA7
Entrée ou Sortie 8 bits
logique A
65
adresse bit 7 6
1 1
Registre du mot de commande
logique de commande
adresse bit 7 6
0 1
Registre B
PB0
PB7
Entrée ou Sortie 8 bits
logique B
adresse bit 7 6
1 0
Registre C
PC0
PC7
Entrée ou Sortie 8 bits
logique C





MOT DE COMMANDE

Attention, il a deux définitions selon que le bit D7 est à 0 ou à 1 :

— D7 à 1

Le mot de commande est conservé dans le registre de commande, et ce, tant qu'on ne le change pas. Il spécifie les modes des fonctionnements des registrs A, B, C.

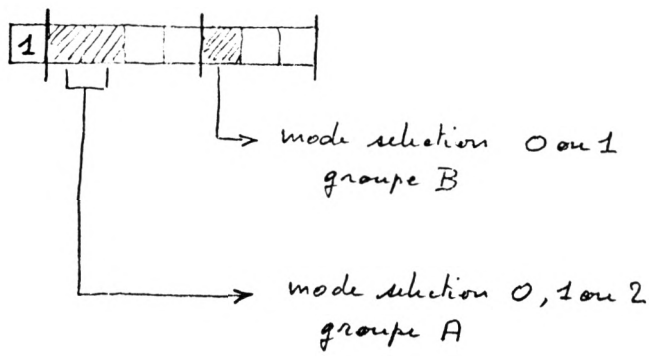
— D7 à 0

Dans ce cas, le mot de commande prend un rôle particulier ; il permet alors de spécifier à 1 ou à 0 l'un des bits du registre C.

Les registres A, B, C peuvent être simultanément utilisés selon les modes différents.

Le mot de commande spécifie simultanément les modes pour :

1 - Registre A Registre C upper	groupe A
2 - Registre B Registre C lower	groupe B



Description détaillée

SELECTION DE MODE - MOT DE COMMANDE

Lors d'un RAZ (par la face avant d'Alcyane, ou à la mise sous tension), les registres A, B, et C sont remis à zéro et désactivés.

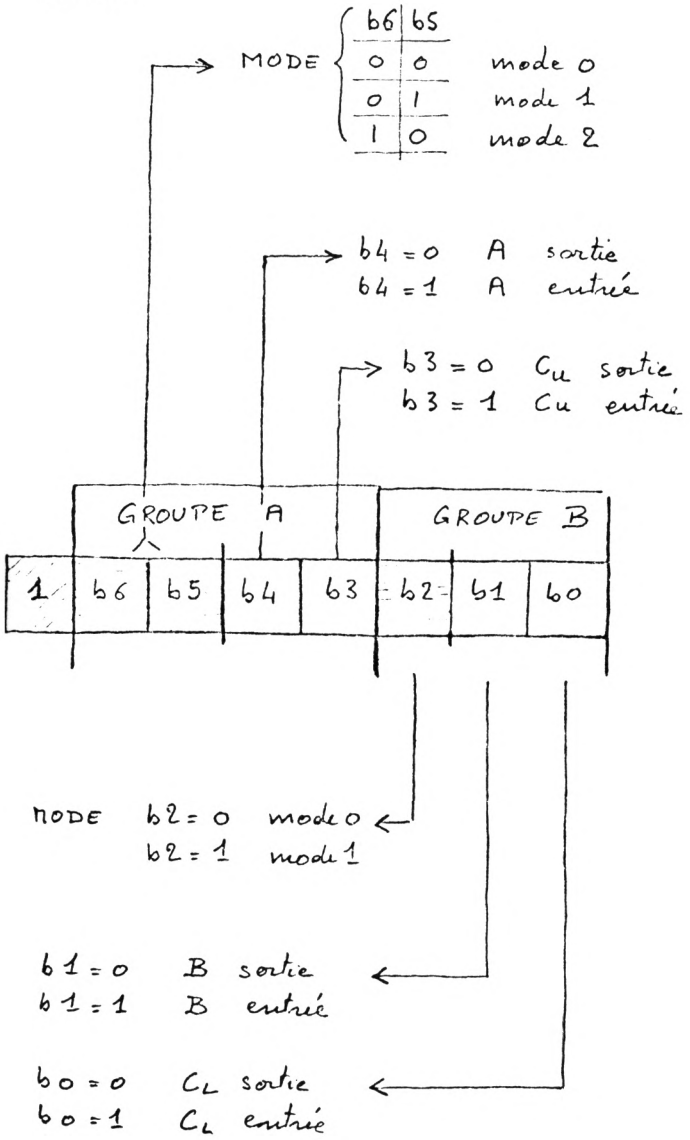
Le mot commande est conservé dans le registre d'état, et n'est effacé que par RAZ ou par l'inscription d'un autre mot de commande.

MISE A ZERO OU A 1 D'UN BIT DE C

Le registre C, comme nous l'avons vu précédemment, et nous l'illustrerons dans des exemples, est souvent employé pour contrôler des lignes de commandes.

Dans ce cas, il est intéressant de pouvoir individuellement sélectionner chacun de ses bits pour les mettre à 1 ou à 0, par programme.

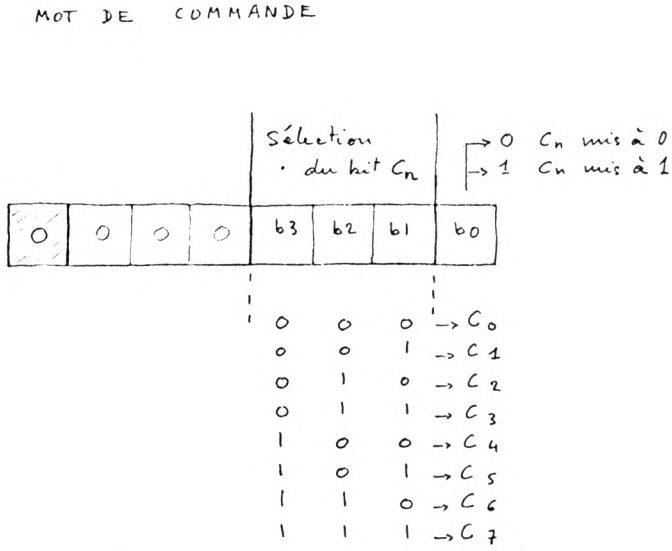
L'opération inverse n'est pas possible. Pour lire l'état d'un bit, il faut lire le registre dans son ensemble.



MODE 0

Les registres A, B, Cu, CL, sont utilisés tels quels.

Les contenus de ces registres seront donc directement transférés dans l'accumulateur pour une



Exemple : mettre le bit C4 à 1, puis C2 à zéro

```
MVI A
09                0 0 0 0 1 0 0 1   mot Cd
OUT
C1                1 1 0 0 0 0 0 1   mot Ad
MVI A
04                0 0 0 0 0 1 0 0   mot Cd
OUT
C1                1 1 0 0 0 0 0 1   mot Ad
```

Entrée, ou le contenu de l'accumulateur directement transféré dans un des registres pour une Sortie.

Deux cas sont à distinguer :

— Sortie

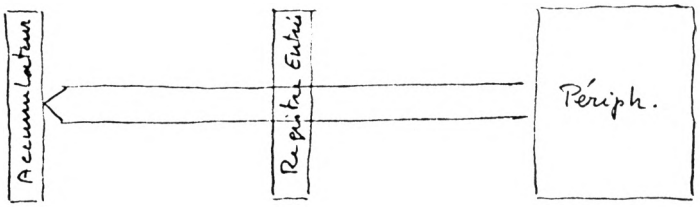
Le mot écrit de l'accumulateur sur le registre de sortie est mémorisé dans ce registre jusqu'à nouvelle écriture.

— Entrée

Le mot lu dans l'accumulateur à travers le registre considéré a strictement la même valeur que les lignes arrivant à ce même registre.

Ceci implique que la valeur de ces lignes doit être stabilisée au moment de l'exécution d'une instruction de lecture.

On peut aussi dire que dans ce cas (MODE 0) les registres d'entrées sont entièrement transparents.



Le mot de commande définit les différentes combinaisons : registre en entrée, ou registre en sortie, de A, B, C_U, C_L.

MODE 1

C'est le mode le plus employé, car il permet d'établir un diaglogue entre le processeur et le périphérique, à peu près comme ceci :

Processeur - J'ai quelque chose à vous transmettre.
Périphérique - Non ! je ne suis pas prêt.
Quelques instants plus tard :
Processeur - J'ai quelque chose à vous transmettre.
Périphérique - OK, je suis prêt.
Périphérique - Ça y est, j'ai pris l'information.
Processeur - J'ai quelque chose à vous transmettre à nouveau...
et ainsi de suite jusqu'à la fin du transfert complet.

En jargon technique, cette méthode est appelée HAND SHAKE (traduction littérale : poignée de main).

Nous verrons que sur l'exemple d'interfaçage de l'imprimante 80 colonnes, nous utilisons un système identique, mais simulé par programme en mode 0, pour plus de souplesse.

REGISTRES A et B en mode 1 ENTREE

Les bits du registre C servent aux lignes de commande pour le contrôle du HAND SHAKE.

Rôle de chacune des lignes :

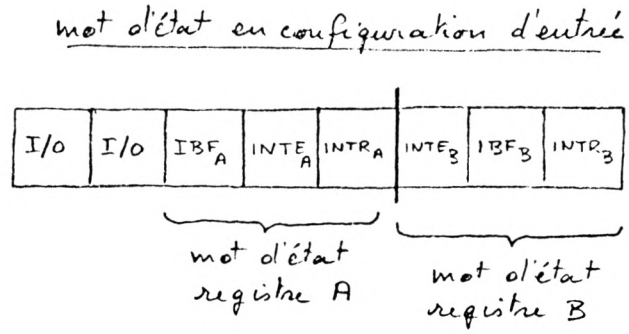
$\overline{STB}$ (strobe input)	actif bas, transfère la valeur des lignes d'entrée A <0:7> ou B <0:7> dans le registre A, ou B (la valeur de ces lignes peut immédiatement après changer).
IBF (input buffet full)	actif haut, indique que le chargement du registre d'entrée a été effectué. Il est mis à 1 par $\overline{STB}$ à 0, et remis à 0 après lecture par le processeur, ($\overline{RD}$), du registre.
INTR (interrupt request)	Demande d'interruption. Cette ligne prend la valeur 1 si simultanément : $\overline{STB}$ = 1, IBF = 1, et que INTE (A ou B) est à 1. Elle retombe à zéro après lecture du registre par le processeur ($\overline{RD}$).
$INTE_A$	est contrôlé par le bit PC4 du registre C.
$INTE_B$	est contrôlé par le bit PC2 du registre C.

Quelques remarques :

— le registre C est pratiquement uniquement employé au contrôle des échanges sur les registres A et B, excepté PC6 et 7. Ces derniers peuvent être mis en entrée ou en sortie selon la valeur du mot commande, partie groupe A. Mais s'ils sont spécifiés en bits de sortie, ils ne pourront être mis à 1 ou à 0 qu'en utilisant le mot de commande permettant de ne positionner qu'un bit à la fois.

$INTE_A$ et $INTE_B$ sont positionnés de la même façon :

— le registre C dans son entier peut être lu par le processeur. Dans ce cas, il devient un mot d'état des entrées/sorties et est susceptible d'être testé par programme.

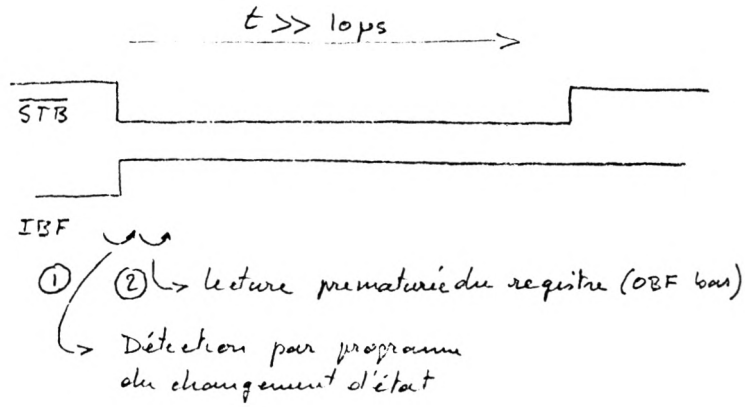


Contrôle par programme

La ligne STB est uniquement contrôlée par le périphérique. Par convention généralement admise, elle reste à l'état bas tant que le périphérique est occupé à exécuter l'action demandée (cette hypothèse est toujours à vérifier, car malheureusement il y a des exceptions).

Puisque IBF est actif dès que STB est actif, il en résulte en général qu'il ne faut pas, par programme, tester l'état de la ligne IBF pour déterminer le moment de lecture du registre. Ceci est surtout vrai si le périphérique est lent avec OBF actif sur plus de quelques microsecondes (voir plus loin OBF).

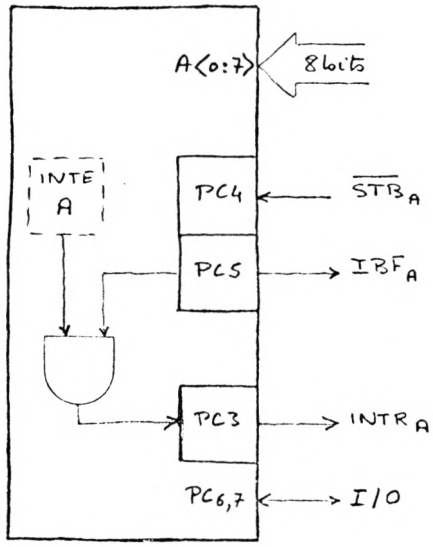
Pour plus de sécurité, employez la ligne INTR qui peut être testée par programme (lecture des bits PC3 ou PC0). Elle est sans ambiguïté.

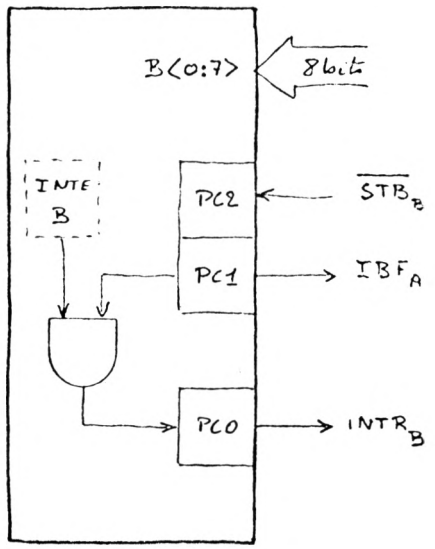


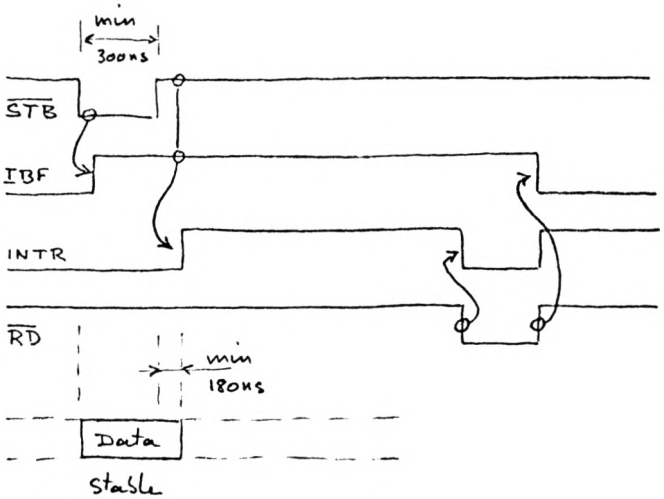
Contrôle par interruption

La ligne INTR est directement reliée à la carte module de gestion des interruptions.

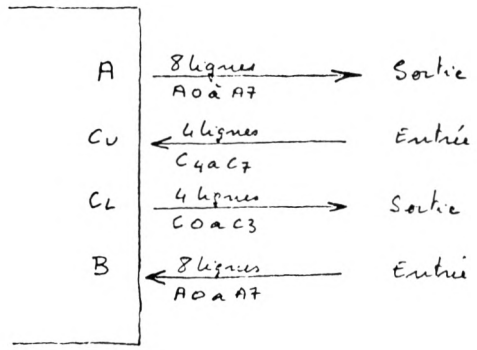
Les différents points que nous venons de voir, s'appliquent également à ce qui suit (mode 1 sortie et mode 2).



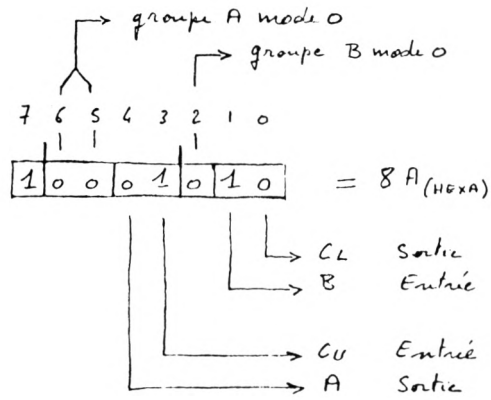




Exemple :



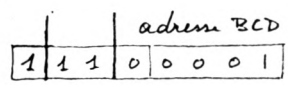
Mot de commande :



Le programme :

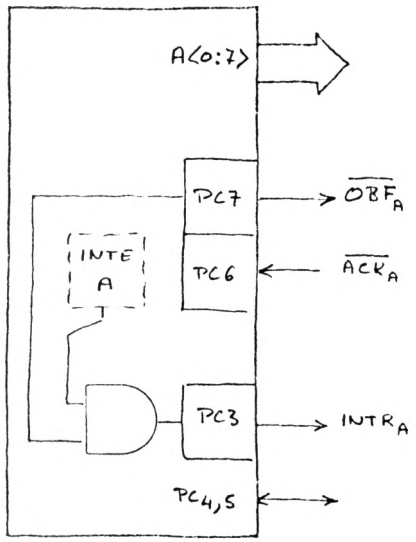
```
MVI  A   } Chargement dans
8A       } l'accumulateur du
OUT      } mot de commande
E1       } Sortie du contenu de
                l'accumulateur sur
                le registre de commande
```

Mot d'adresse périph.
E1 pour :

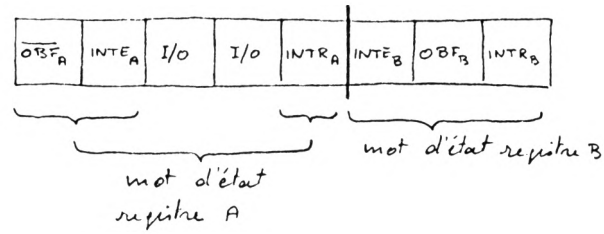


systématique sur mot d'adresse périphérique

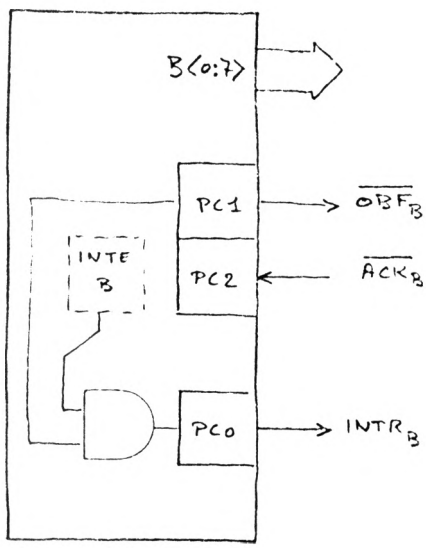
adresse du registre mot de commande

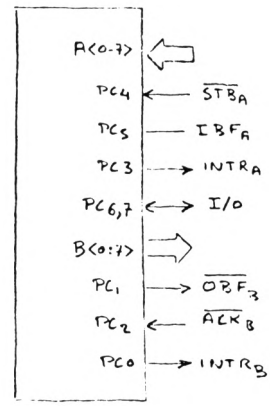


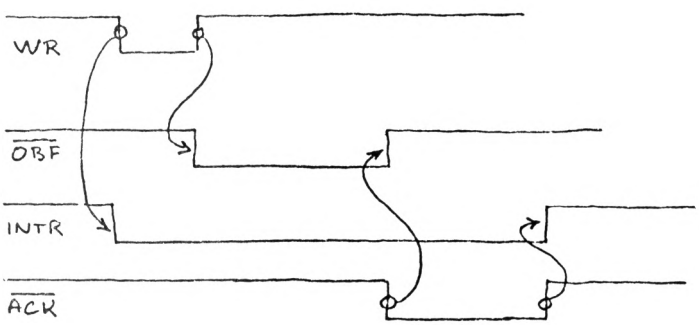
Mot d'état (lecture du registre C) en configuration Sortie

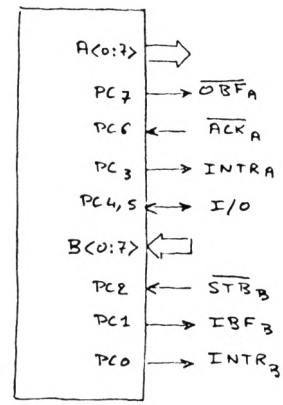


COMBINAISONS POSSIBLES









REGISTRES A et B en mode 1 SORTIE

Le fonctionnement est très similaire au mode 1 entrée :

$\overline{OBF}$ (output buffer full) actif bas, indique au périphérique que le processeur a rempli le registre de sortie.

$\overline{ACK}$ (acknowledge input) actif bas, provenant du périphérique, indique que celui-ci a accepté les données, et est disponible pour un nouveau transfert.

INTR actif haut pour demande d'interruption, mis à 1 pour : $\overline{ACK}$ = 1, $\overline{OBF}$ = 1, remis à 0 après écriture d'une nouvelle valeur sur le registre de sortie, par le processeur ($\overline{WR}$).

$INTE_A$ est contrôlé par le bit PC6 du registre C.

$INTE_B$ est contrôlé par le bit PC2 du registre C.

MODE 2

Dans ce mode (encore peu employé) les informations vont dans un sens ou dans un autre, sur les mêmes lignes (ceci fait appel à une technologie toute récente des circuits intégrés).

Seul le registre A possède cette propriété. Le périphérique doit être couplé à la liaison bi-directionnelle par l'intermédiaire de registres 3 états.

INTR (demande d'interruption) actif haut, est utilisé aussi bien pour une opération d'entrée que de sortie.

Opération de sortie

$\overline{OBF}$ (actif bas) égal à 0 pour indiquer que le processeur a écrit un mot dans le registre de sortie A.

$\overline{ACK}$ (actif bas) égal à 0, active la sortie 3 états du registre A ; $\overline{ACK}$ = 1, la sortie 3 états est mise à haute impédance.

$INTE_1$ (associé à OBF) est contrôlé par le bit PC6.

Opération d'entrée

$\overline{STB}$ (actif bas) transfère la valeur des lignes d'entrée sur A<0:7>

IBF (actif haut) indique que le registre d'entrée a été rempli.

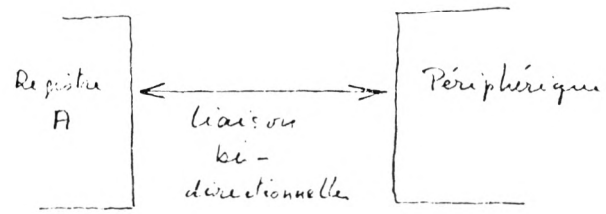
$INTE_2$ (associé à IBF) est contrôlé par le bit PC4.

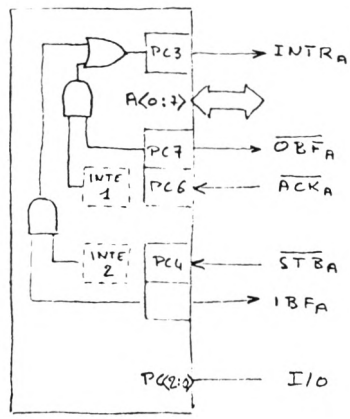
Utilisation pratique de la carte

Cette carte est appelée "universelle", car elle possède un potentiel intrinsèque important. Pour concrétiser son potentiel dans une application donnée, il faut cependant spécialiser les lignes de sortie des registres A, B et C.

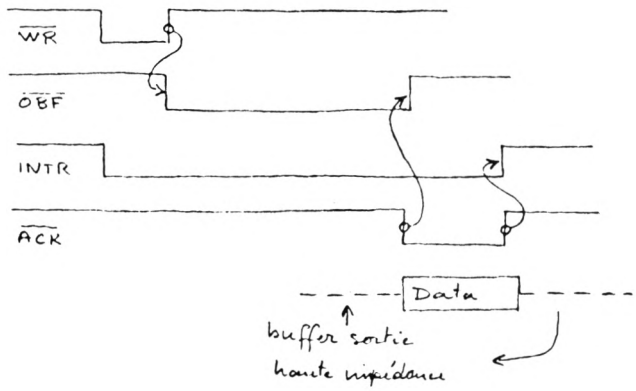
De plus, ces lignes de sortie proviennent d'un circuit intégré en technologie MOS incapable d'absorber beaucoup de courant (1 charge TTL, soit 1,6 mA). Il faut donc généralement prévoir une adaptation de puissance.

Pour toutes ces raisons, la carte a un format environ 2/3 du format standard. Ceci rend possible

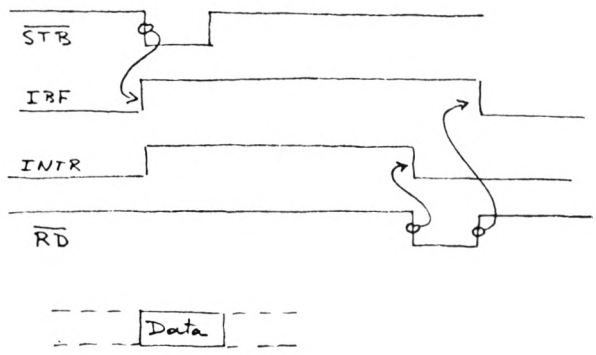




Sortie



Entrée



l'adjonction, en prolongement, d'un circuit imprimé au format 1/3 de carte pour adaptation tant logique qu'électrique de puissance.

Exemple :

— 4002-1 Carte module interface Imprimante MANNESMANN :

— 4002-1 Carte module interface 8 bits E/S pour BASIC.

Module d'entrée/sortie 8 bits Basic

C'est une carte d'entrée/sortie 8 bits parallèle qui comprend les signaux de commande pour cadencement du dialogue d'entrée/sortie (HAND SHAKE).

Elle est principalement destinée à être utilisée avec le langage BASIC d'Alcyane.

Principe de fonctionnement

Convention

Le mot SORTIE décrit un transfert d'information de l'unité centrale, vers un périphérique, et le mot ENTREE, un transfert d'information du périphérique vers l'unité centrale.

Le module comprend deux registres de 8 bits et quatre lignes de HAND SHAKE.

— Registre A - en SORTIE

Il mémorise l'information à transférer sur un périphérique. Il lui est associé la ligne de commande OBF pour signaler au périphérique que A est prêt à lui transmettre une information ; et la ligne ACK, commandée par le périphérique, pour signaler que ce dernier a pris l'information.

— Registre B - en ENTREE

Accepte, et mémorise l'information envoyée par le périphérique. De même que pour le registre A, il lui est associé la ligne STB, commandée par le périphérique, pour signaler que celui-ci vient d'envoyer un caractère, et la ligne IBF, qui sert à la réponse du module E/S pour dire que ce caractère est en cours d'acceptation.

Caractéristiques électriques

LIGNE DE SORTIE A0 à 07, OBF, IBF

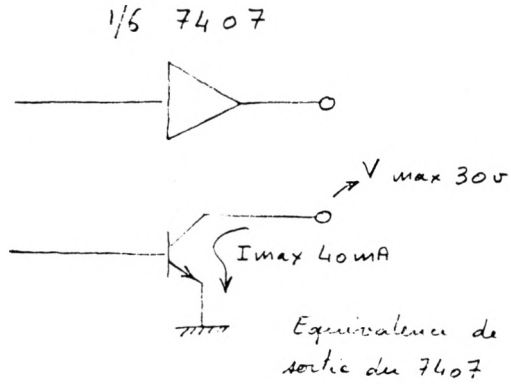
Chaque ligne de sortie (8 + 2) est à collecteur ouvert non inverseur.

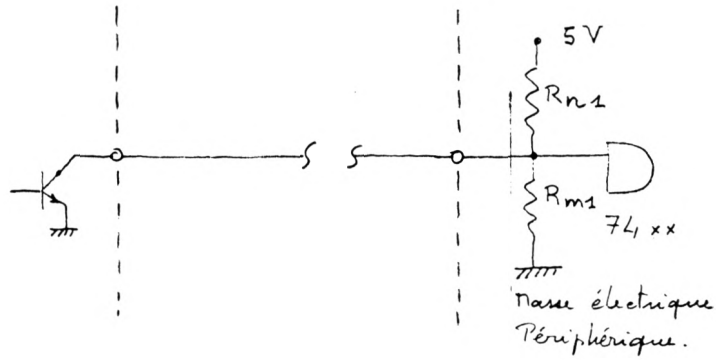
Pour une liaison mode E/S périphérique par câble plat, il est recommandé :

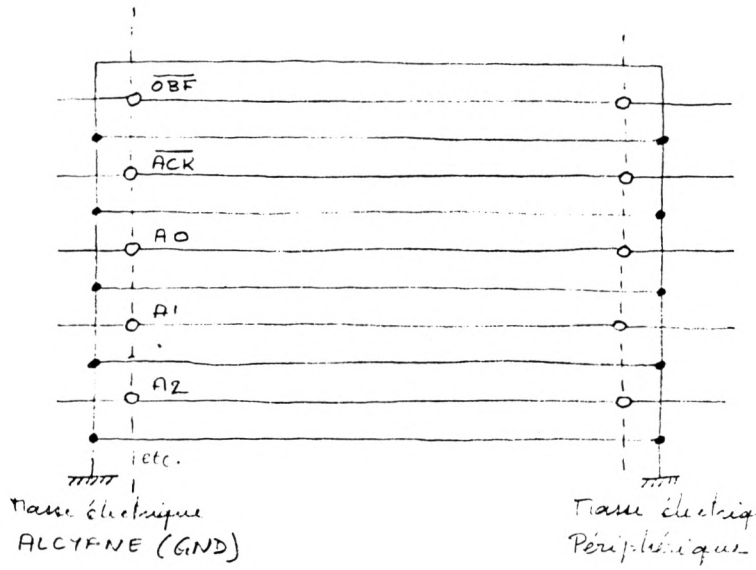
— de terminer la ligne par un pont résistif,

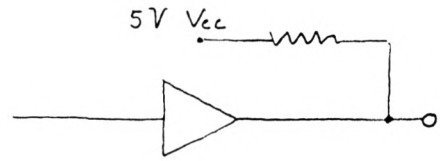
— d'utiliser un conducteur sur deux (câble plat) en masse électrique.

Pour de courtes distances dans des lieux non parasités, ces précautions ne sont pas toujours utiles. Dans ce cas, vous pouvez remplacer le pont résistif Rn_1/Rm_1 par une seule résistance (voir schéma).



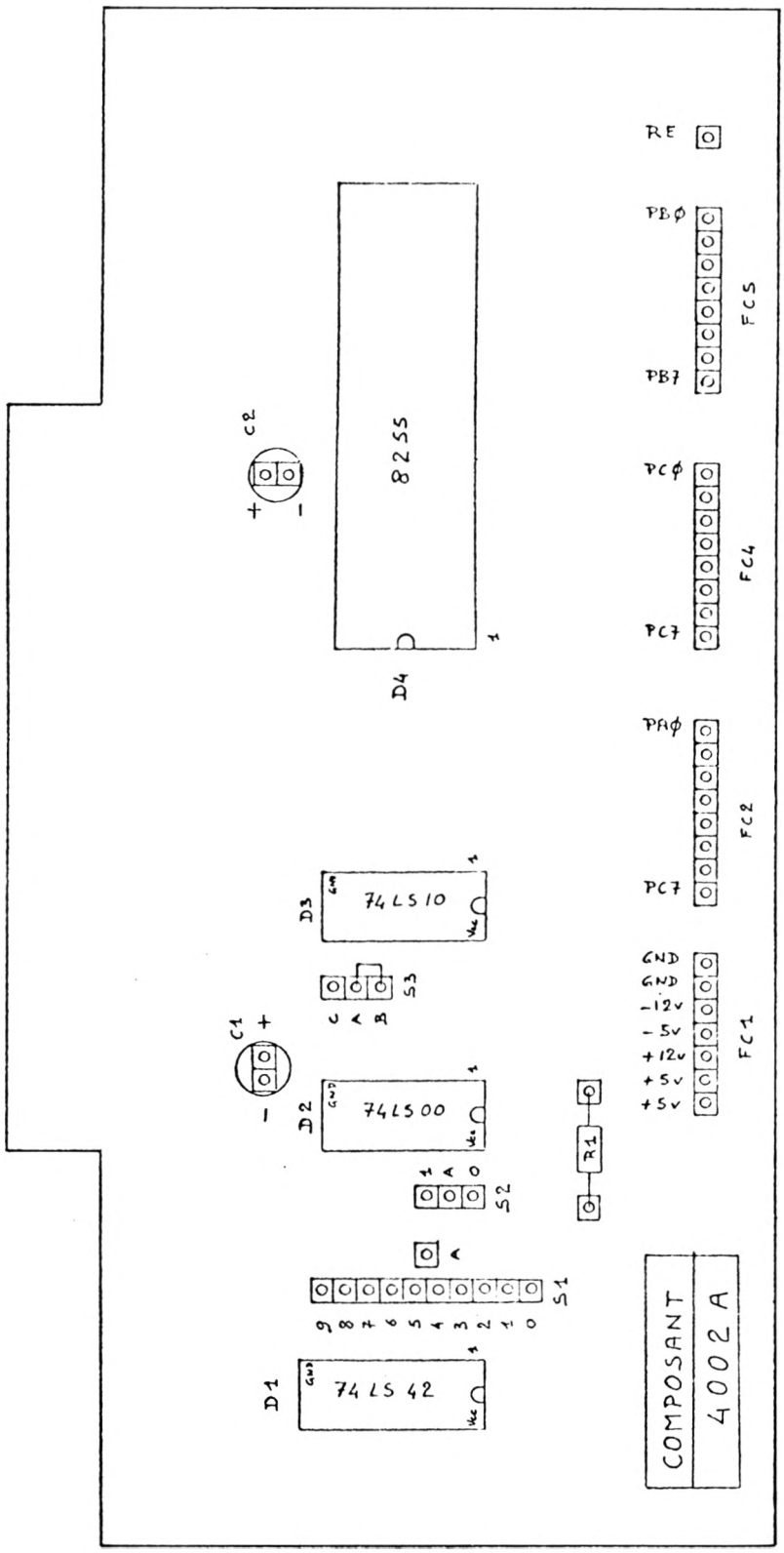






CARTE INTERFACE PARALLÈLE UNIVERSEL PROGRAMMABLE

4002 A



C1 tantale goutte 2,2µf
C2 " " "

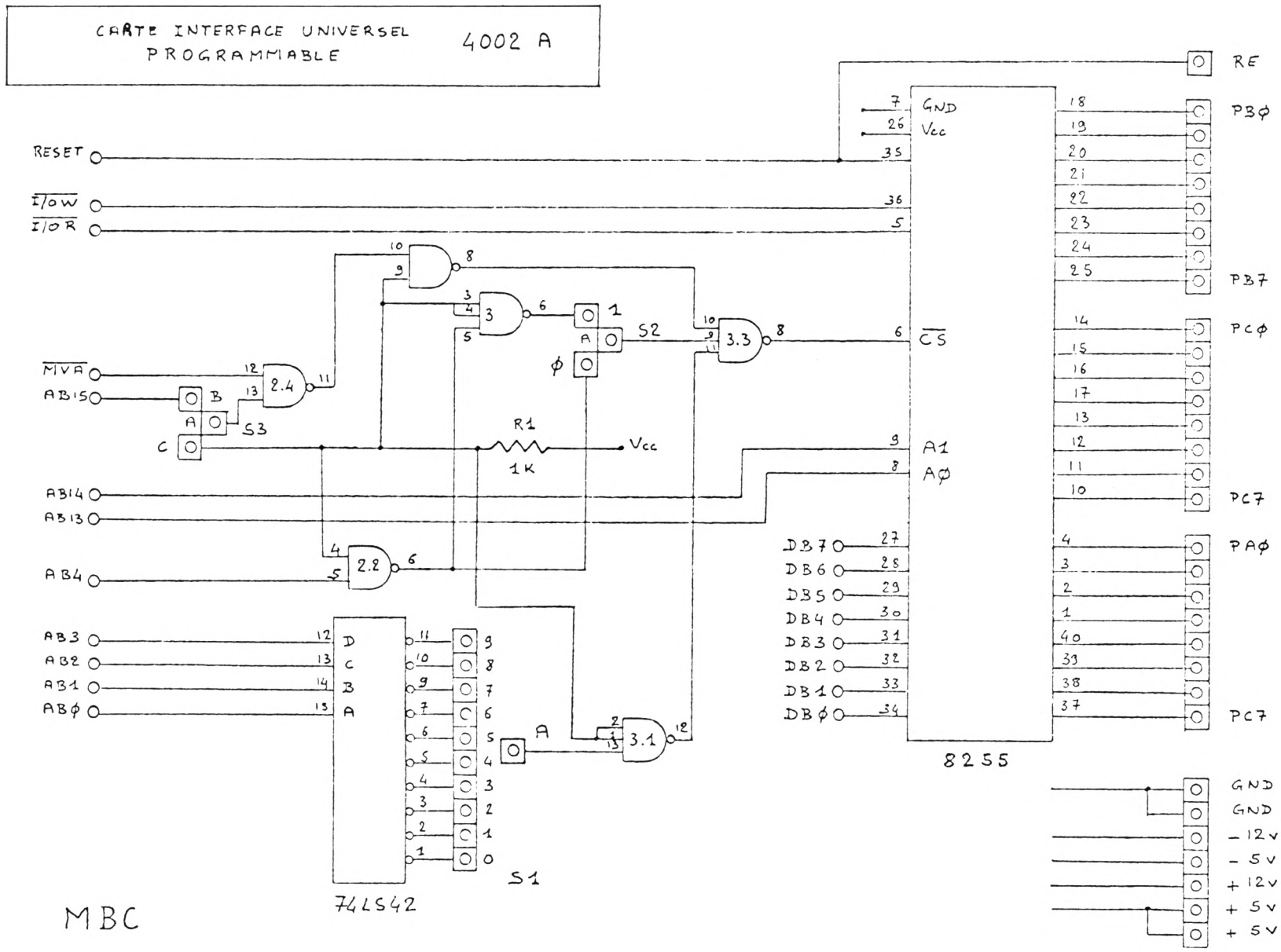
R1 1 KΩ

S1 unité de l'adresse (BCD)

S2 dizaine de l'adresse (BCD

(relier le plot A au plot correspondant à l'adresse voulue)

S3 sauf spécification particulière relier A à B

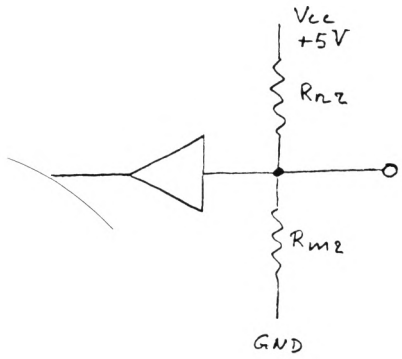

CARTE INTERFACE UNIVERSEL PROGRAMMABLE
4002 A
RESET
I/OW
I/OR
MVA
AB15
AB14
AB13
AB4
AB3
AB2
AB1
ABØ
R1
1K
Vcc
S1
S2
S3
74LS42
8255
GND
Vcc
CS
A1
AØ
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DBØ
RE
PBØ
PB7
PCØ
PC7
PAØ
PC7
GND
GND
- 12v
- 5v
+ 12v
+ 5v
+ 5v
MBC

LIGNES D'ENTREES B0 à B7, ACK, STB

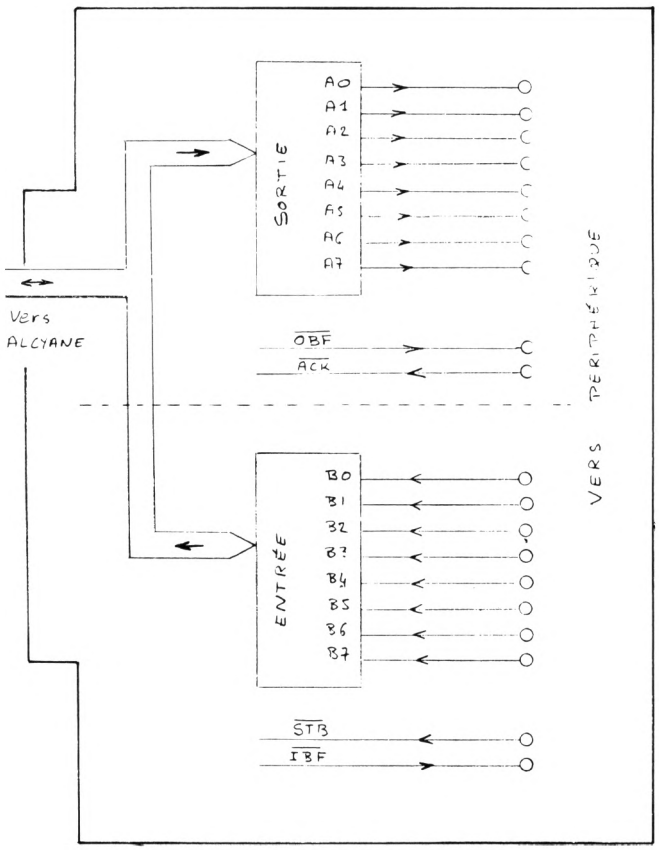
Chaque ligne arrive sur un pont résistif, puis sur une porte TTL.

Le pont résistif est prévu pour des "drivers" de ligne du type 7407.

De même, pour des liaisons courtes, il est éventuellement possible de supprimer ce pont, ou bien d'augmenter les valeurs des résistances Rn_2 et Rm_2.



DESCRIPTION FONCTIONNELLE



Il est très recommandé d'utiliser les mêmes précautions d'alternance lignes de signaux, lignes de masse, sur le câble plat.

Lignes de commande

EN SORTIE (REGISTRE A)

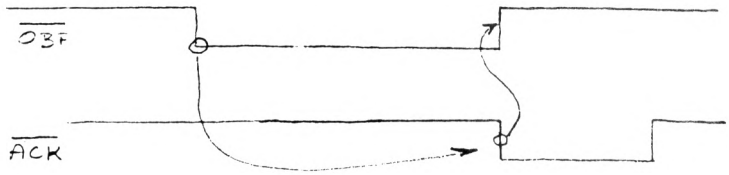
$\overline{OBF}$

Cette ligne signale au périphérique que le registre de sortie contient une information valide.

$\overline{ACK}$

Est une ligne, contrôlée par le périphérique en réponse à $\overline{OBF}$ pour signaler qu'il vient de prendre l'information.

L'information contenue dans le registre de sortie est disponible dès le flanc tombant de OBF.



EN ENTREE

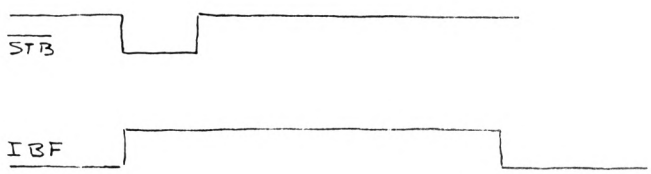
IBF (contrôlé par la carte interface 4002)

IBF = 0 indique au périphérique qu'il peut présenter son information sur les lignes d'entrée du registre B, tout en émettant simultanément un signal de commande STB pour indiquer que les lignes d'entrées sont valides.

Sur le flanc tombant de STB, IBF remonte.

IBF reste haut jusqu'à lecture, par le processeur du registre d'entrée.

Nota : les informations d'entrée *sont* mémorisées dans le registre B sur le flanc montant de STB, il n'est donc pas nécessaire de maintenir les lignes d'entrées stables pendant toute la durée de IBF.



7 - 9 - 3. Entrées/sorties série

C'est un sujet assez vaste dont nous n'aborderons que les domaines qui peuvent être utiles pour une utilisation immédiate, ou presque... avec Alcyane.

Nous étudierons donc les points suivants :

— structure de l'information en transmission série asynchrone,
— le standard CCITT/RS232C liaison asynchrone,
— utilisation de la carte 4021 en mode simulation de modem.

Ces clarifications seront utiles, étant donné l'importance et parfois la complexité de ce mode privilégié de transmission à courte, moyenne et longue distance.

Structure de l'information

Le code le plus communément utilisé est l'ASCII avec ou sans parité. C'est un code à huit bits, dont sept bits d'information utile. Le huitième bit est mis à 1 ou à 0 selon que la somme des sept autres est paire ou impaire, de sorte que la somme de l'ensemble des huit bits soit toujours paire, ou toujours impaire.

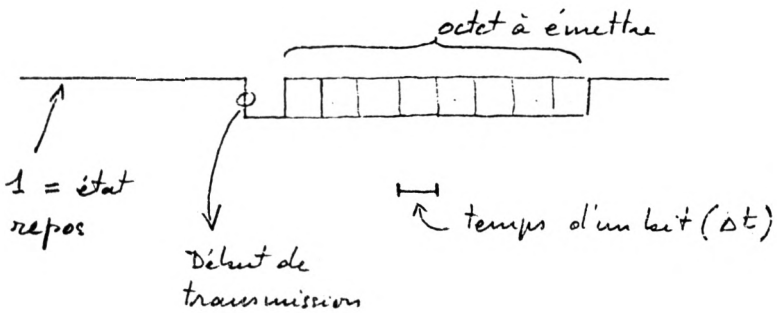
Pour que le récepteur puisse reconnaître l'information transmise sur une ligne, il lui est nécessaire de se synchroniser sur la vitesse de transmission de l'émetteur.

Il n'existe que deux méthodes.

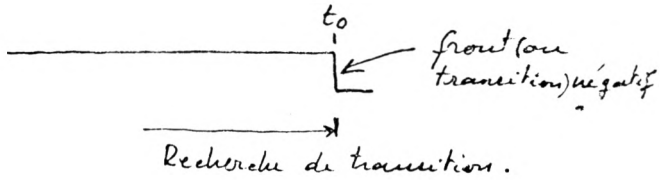
Liaison asynchrone

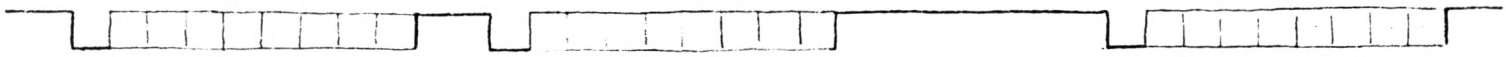
Emetteur et récepteur sont réglés à la même vitesse de transmission. Mais chaque octet de huit bits peut être émis à tout moment, isolé ou en paquet de longueur quelconque.

Au repos, c'est-à-dire en l'absence de tout échange d'information, la ligne est maintenue à l'état 1. Pour transmettre un octet, l'émeteur met la ligne à l'état 0, et ce pendant le temps que représente un bit, puis il transmet l'octet.

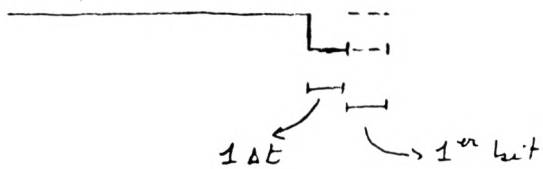


Dans ces conditions, le récepteur sait que dès que la ligne passe de l'état 2 à l'état 0, un octet de huit bits suit. Il est donc obligé de regarder en permanence cette ligne pour détecter une éventuelle transition (dite négative).

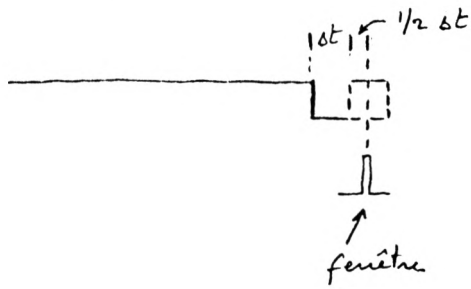




A la transition, il sait également, et par hypothèse, qu'un temps élémentaire de bit plus loin (1 st) apparaît le premier bit d'information.

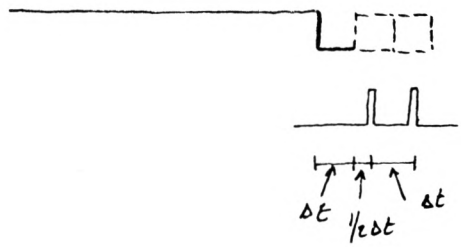


Il suffit donc d'ouvrir une fenêtre au milieu de ce bit, pour savoir s'il est à 0 ou à 1.



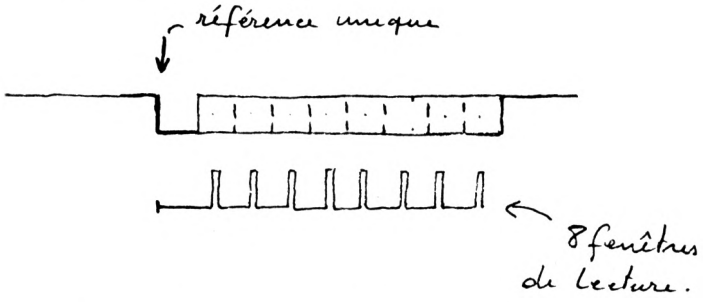
Cette fenêtre est placée dans le temps à 1 st + 1/2 st.

Il en est de même pour le bit suivant, dont le milieu est référencé 1 st + 1/2 st + 1 st plus loin que le front négatif.



En continuant ainsi le raisonnement de proche en proche, le récepteur est capable de relire tous les bits de l'octet transmis.

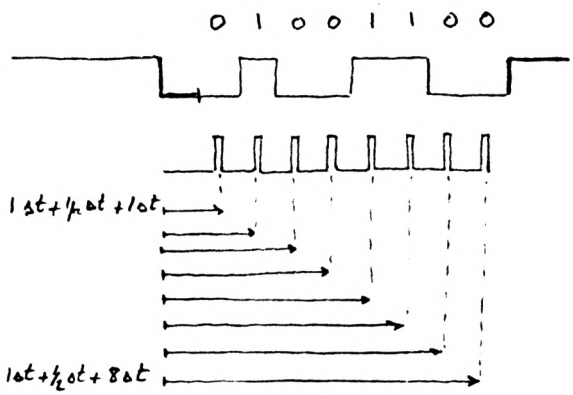
La position relative de chaque fenêtre est seulement connue par rapport à la référence unique que représente la première transition négative, et ce pour tout l'octet transmis.



Pour que le processus puisse recommencer, il faut bien sûr qu'à la fin du dernier bit, la ligne soit remise à l'état 1. Cet état peut durer très longtemps si les octets transmis sont très séparés les uns des autres.

Au contraire, s'ils sont à la suite les uns des autres, la ligne doit rester un temps minimum à l'état 1 pour que le récepteur puisse lui-même se remettre dans les conditions nécessaires à la recherche d'un front descendant.

Ce temps d'attente minimum entre chaque octet reçu est normalement utilisé à ranger cet



octet en mémoire ; il dure en général le temps équivalent d'un ou de deux bits (1 ou 2 st).

Dans cette méthode de transmission, nous voyons que chaque octet n'est pas synchrone d'une base de temps préfixée (Horloge), puisqu'il peut arriver n'importe quand. Par contre, chaque bit à l'intérieur de l'octet est synchrone, puisqu'il est référencé par rapport à un événement : la première transition négative.

Code à 11 moments

Dans l'exemple étudié précédemment, nous avons vu que le premier bit d'information arrive 1 st après le premier front négatif. Et d'autre part, en fin de transmission de l'octet, la ligne doit rester l'équivalent de 2 st à l'état 1.

La durée de transmission d'un octet représente en fait non pas la durée de transmission de huit bits, mais celle de 11 bits. Pour éviter toute confusion, on appelle *moment* un temps élémentaire de transmission que nous avons représenté jusqu'ici par le symbole st.

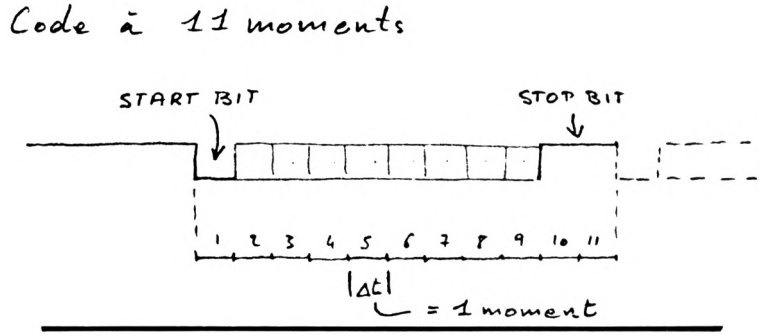
Le premier moment est appelé START BIT, les dix et onzième, STOP BIT.

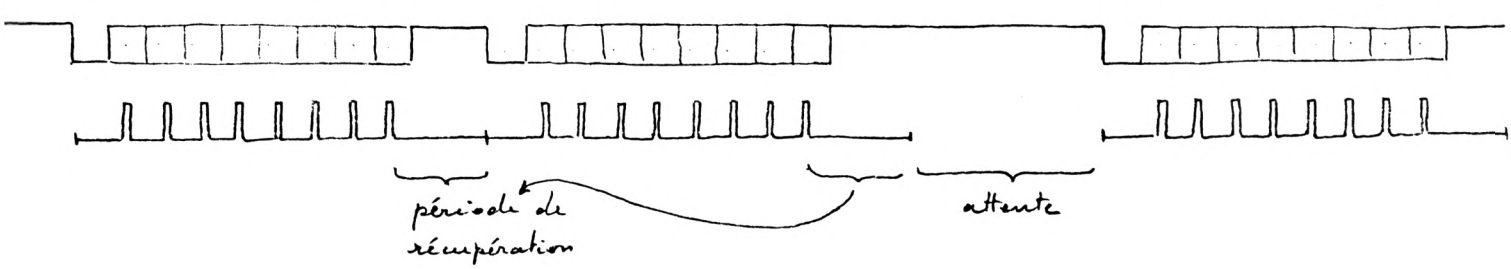
Vitesse de transmission

La vitesse de transmission de l'information sur une ligne est donnée par le nombre maximum de moments transmis par seconde. L'unité est le baud.

La méthode de transmission *asynchrone* est utilisée pour des liaisons relativement lentes, jusqu'à 1200 bauds en général. Dans certains cas très particuliers, on peut éventuellement aller bien au-delà (jusqu'à 50.000 bauds...).

A partir de 2400 bauds, il est préférable de changer de méthode pour de simples raisons de fiabilité de transmission. Dans ce cas, on utilise une liaison *synchrone*. Et c'est une toute autre histoire.

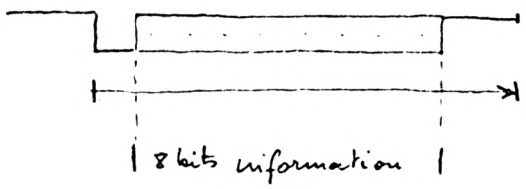




Vitesses normalisées

110 bauds :

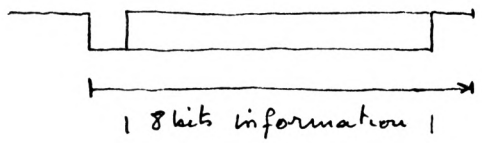
Le code comprend 1 bit *start* et 2 bits *stop* (minimum).



La vitesse réelle de transmission est donc en fait de 10 caractères/seconde.

300 bauds :

Le code comprend 1 bit *start* et 1 bit *stop* (minimum).



Vitesse de transmission : 30 caractères/seconde.

600 bauds - 1200 bauds :

Même structure que le 300 bauds.

2400 bauds - 9600 bauds :

Vitesse non normalisée en liaison *asynchrone.*

Ces deux dernières vitesses de transmission sont malgré tout quelquefois utilisées sur certains périphériques en liaison *asynchrone* locale.

Liaison synchrone

La méthode de reconnaissance des bits d'information dans l'ensemble du message est complètement différente.

Nous n'en dirons que quelques mots, le sujet étant fort complexe.

Le premier principe est que chaque message transmis sur la ligne est constitué d'un nombre connu et toujours le même d'octets.

D'autre part, la reconnaissance de chaque bit est faite en synchronisme avec une horloge à fréquence connue de part et d'autre.

Dans ces conditions, il est nécessaire de donner une structure figée au message, en le complétant par des octets dont le rôle est de décrire la nature de chaque bloc d'information.

La liaison *synchrone* est toujours utilisée pour les transmissions longue distance (réseau PTT) à partir de 2400 bauds.

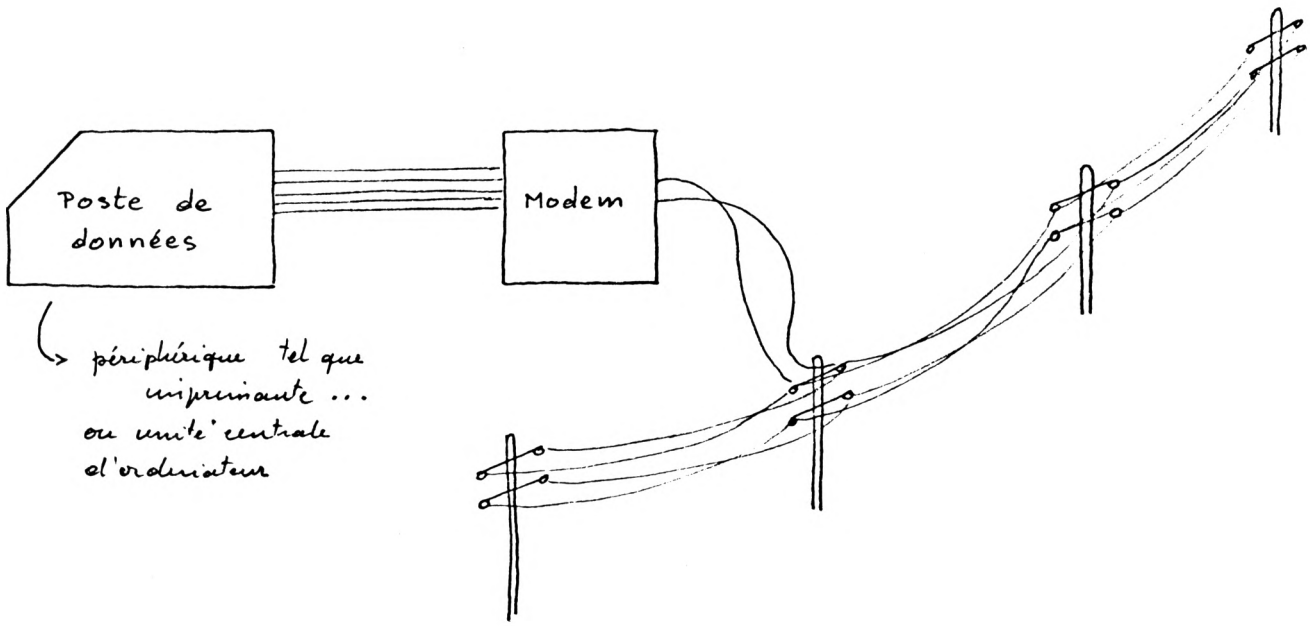
Le standard CCITT V24

Il est appelé souvent simplement V24.

C'est un standard de communication *asynchrone,* le plus anciennement employé, et actuellement, le plus répandu en informatique (très voisin du RS 232 C).

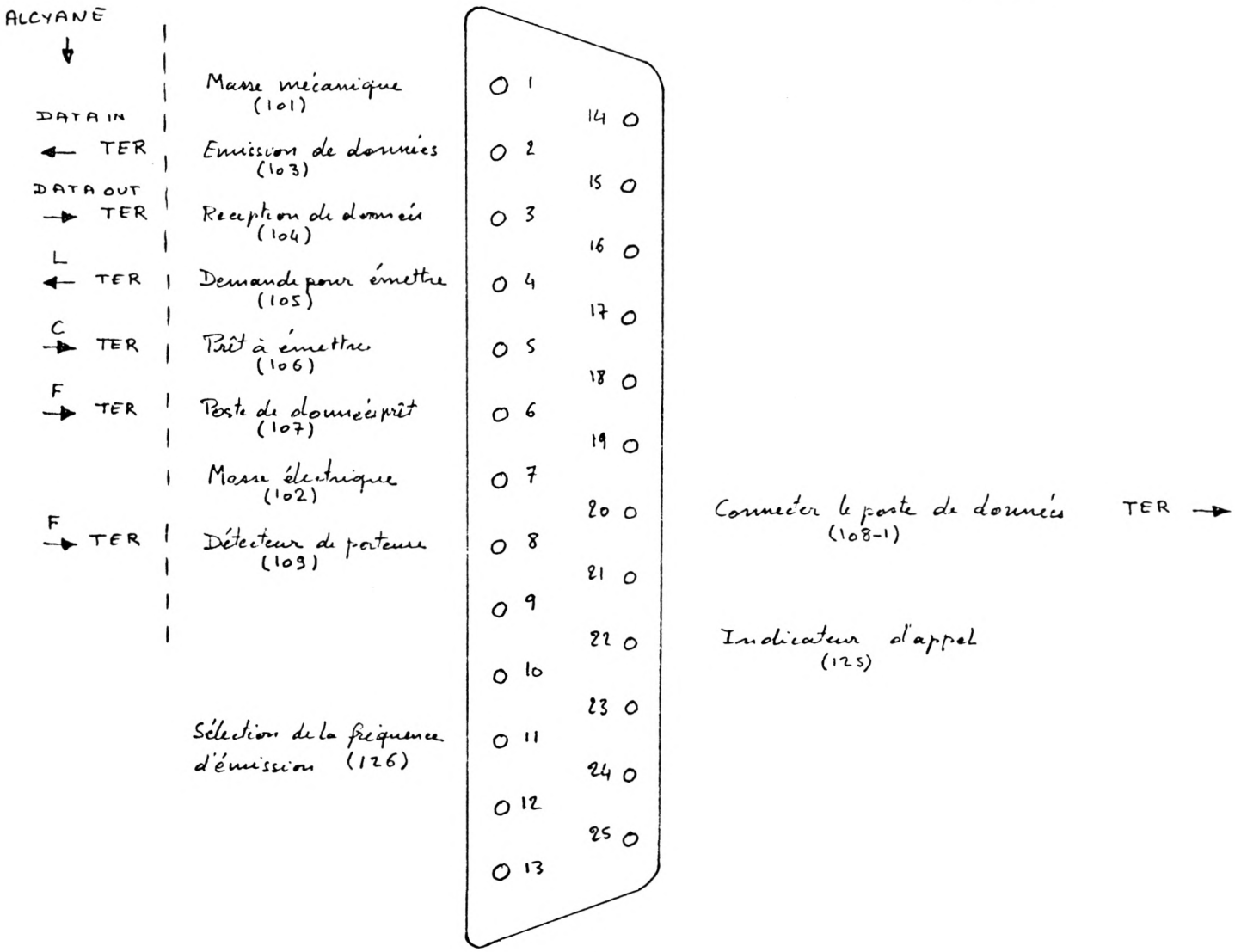
Son premier objet est de permettre de transmettre, par l'intermédiaire d'un *modem*, de l'information binaire sur le réseau téléphonique (PTT ou privé).

Le *modem* est un appareil électronique dont le rôle est de transformer les impulsions électriques, support de l'information binaire, en signaux électriques compatibles avec une ligne téléphonique.

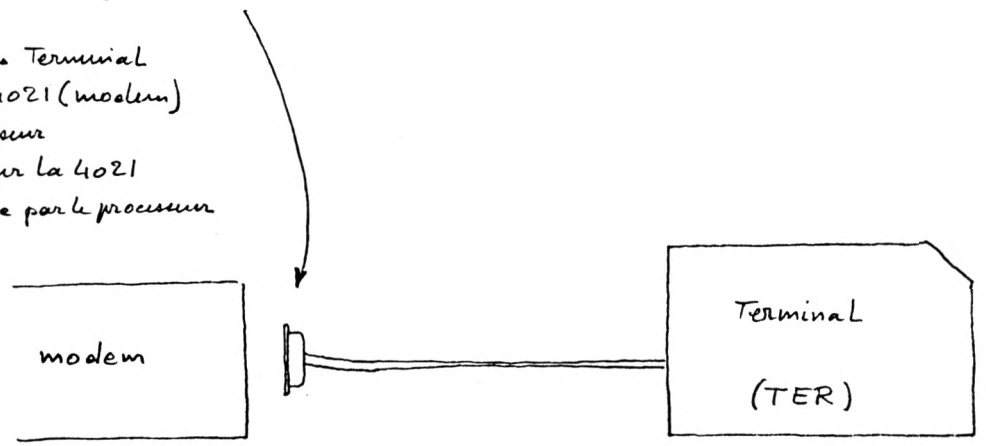


CONNECTEUR NORMALISÉ CCITT
Avis V 24

110	bauds
200	"
300	"
600	"
1200	"

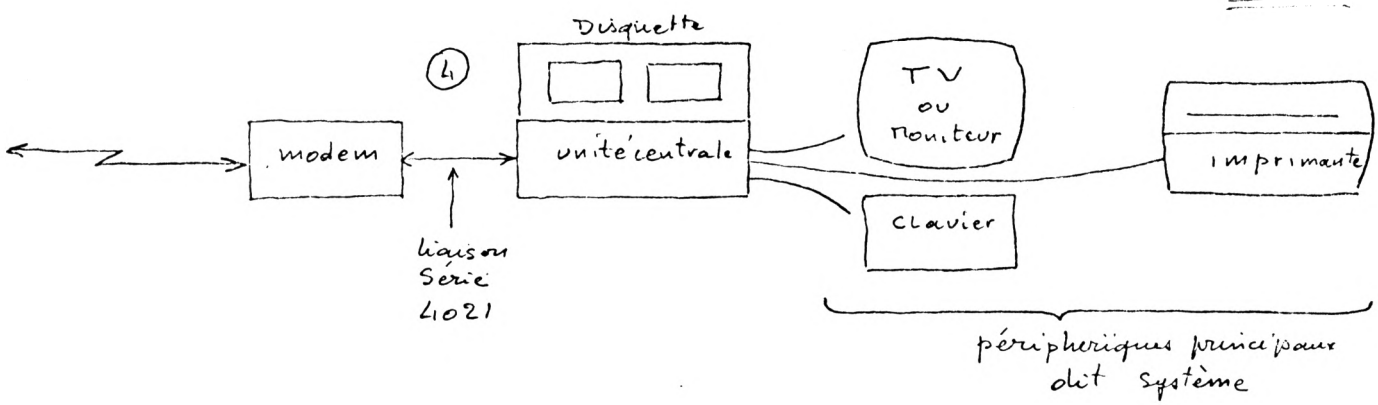


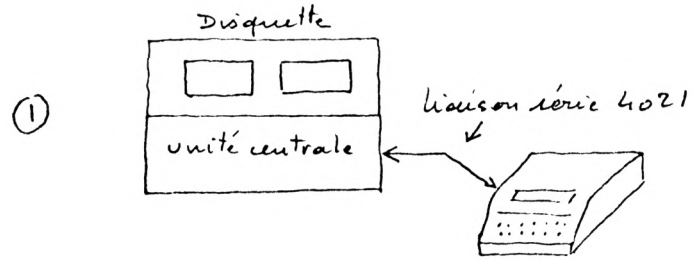
→ TER	4021 (modem) vers Terminal
← TER	Terminal vers 4021 (modem)
L	lu par le processeur
F	Tension fixée sur la 4021
C	Commande émise par le processeur

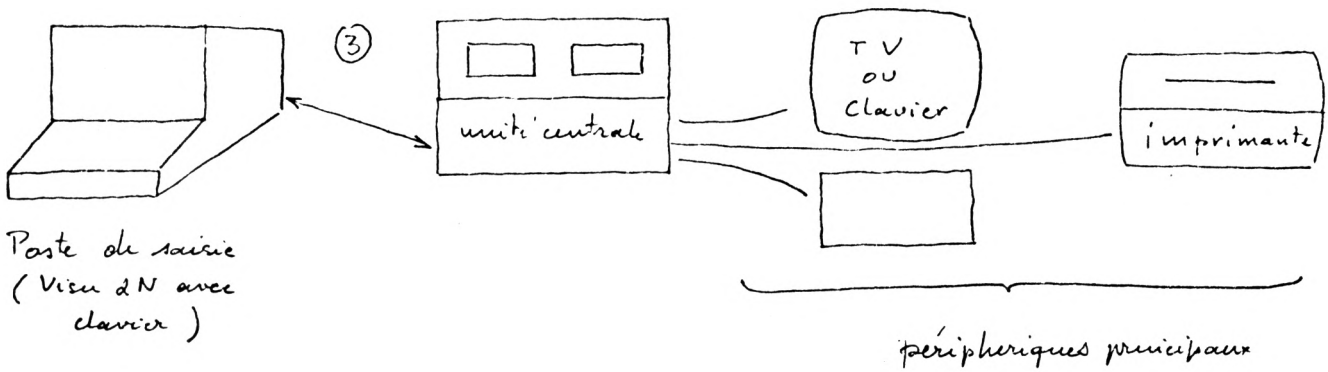


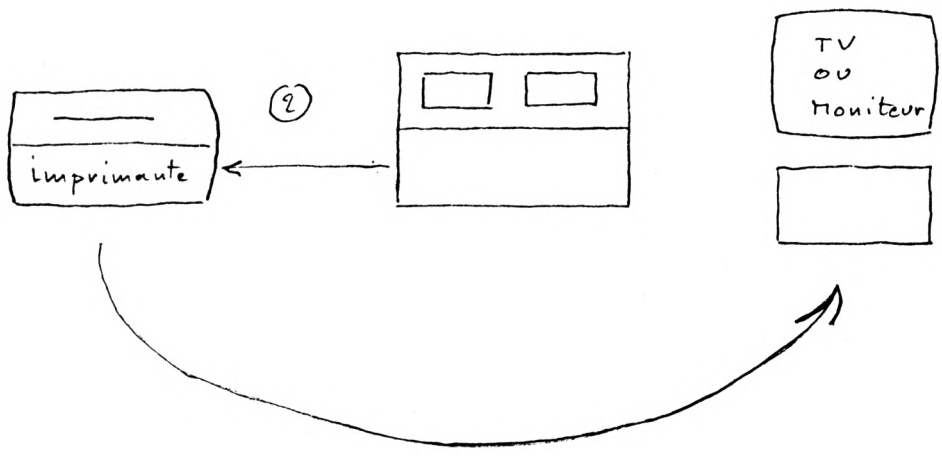
LES DEUX TYPES DE PERIPHERIQUES EN UTILISATION BASIC

Prérit









Les deux formes de codage, modulation de fréquence et modulation de phase, sont employées selon les vitesses de transmission utilisées.

Mais ne nous éloignons pas trop du sujet ! Seule la connaissance des signaux utilisés entre un poste de données et le modem nous est nécessaire pour utiliser à bon escient la carte entrée/sortie série d'Alcyane.

Les lignes

Chaque ligne (de commande ou de données) reçoit un numéro défini dans le standard, et est reliée à une prise normalisée de 25 broches. Chaque broche, ou picot, possède également un numéro.

Lignes utilisées pour une liaison asynchrone :

— 101 picot 1 : Ligne destinée à relier les masses mécaniques entre elles (pour des raisons de sécurité).
— 102 picot 7 : Signal de référence des tensions électriques (GND ou masse électrique sur Alcyane).
— 103 picot 2 : Emission de données, dans le sens poste de données à modem.

Il faut faire très attention aux conventions utilisées. Dans le cas où l'on relie un périphérique équipé d'un connecteur V 24 à Alcyane, "Emission de données" correspond à une opération d'entrée (IN). Mais dans le cas contraire où Alcyane est reliée à un modem, "Emission de données" correspond cette fois à une opération de sortie (OUT).

Emission de donnée est, dans ce standard, relatif à celui qui émet réellement les données vers le modem (ou son substitut, Alcyane, qui peut le simuler : cas d'un périphérique relié par la carte 4021). Il en est de même pour la ligne suivante.

— 104 picot 3 : Réception des données.
— 105 picot 4 : Demande pour émettre. Ce signal est issu du poste de données pour signaler au modem qu'il est prêt à émettre. Le modem signale à son tour, en réponse à la demande pour émettre, par la ligne 106, "Prêt à émettre".
— 106 picot 5 : Prêt à émettre, ce signal issu du modem, est la réponse de ce dernier à la ligne 105 "Demande pour émettre".

Les lignes 105 et 106 ne sont pas toujours utilisées (exemple : modem 200 bauds TRADAN 1100 de LTT, mais sont par contre utilisées sur le 600/1200 bauds TRADAN 1200).

— 107 picot 6 : "modem prêt". Ce signal est fourni par le modem pour signaler au poste de données qu'il est branché sur la ligne téléphonique.
— 108/1 picot 20 : Emis par le poste de données, il provoque la connexion du modem à la ligne téléphonique.
— 109 picot 8 : "Détection de porteuse". Emis par le modem, il indique au poste de données la bonne réception de la fréquence porteuse venant de la ligne téléphonique.
— 125 picot 22 : "Indicateur d'appel". Emis par le modem, il indique la présence d'un appel téléphonique sur la ligne.
— 126 picot 11 : "Sélection de la fréquence d'émission".

Convention de tension

Le "1" logique est appelé "Etat travail".

Le "0" logique est appelé "Etat repos". (C'est là le langage des transmetteurs, bien plus ancien que celui des informaticiens...).

L'état travail est défini par une tension comprise entre + 5 v et + 12 v, l'état repos par une tension comprise entre — 5 v et — 12 v.

Lorsqu'une des conditions décrites plus haut (lignes 101 à 126...) est vérifiée, sa ligne est à l'état travail.

Il en est de même pour les données. S'il n'y a rien sur la ligne téléphonique (pas de données entre deux émissions de données), les lignes 103 et, ou 104 sont à l'état repos (entre — 12 v et — 5v).

Le bit start correspond à l'état travail, le, ou les bits stop, à l'état repos.

Emploi du module série

Chacun des bits transmis est généré par le processeur. De même, chacun des bits reçu est reconstitué par le processeur.

Ceci implique deux règles :
— le processeur est entièrement occupé à gérer la ligne de transmission lors de l'échange d'un message,
— si le processeur est occupé à exécuter une autre tâche, toute transmission en réception de la ligne est perdue.

Nota

En Basic, on distingue deux types de périphériques :

1. Ceux implicitement utilisés par Basic (ou un autre langage). Ce sont les "périphériques système" qui ont des ordres particuliers tels que :

PRINT
DISPLAY
INPUT

2. Les périphériques normaux commandés par des ordres tels que :

WRITE
READ

(voir planche).

Le module d'entrées/sorties, série

Préliminaire

Ce module permet de relier à Alcyane tout périphérique à entrée-sortie série, asynchrone et généralement full duplex,
— en boucle de courant,
— en tension au standard V 24.

D'autre part Alcyane peut être reliée à une ligne téléphonique, par l'intermédiaire d'un modem, par cette même carte (asynchrone full duplex).

Le standard V 24 normalise les échanges entre un poste de données (unité centrale, ou périphérique) et un modem. Mais souvent seules quelques lignes de dialogue parmi les 24 sont utilisées.

La software de transmission, ainsi que le module série couvrent la majeure partie des centaines de variantes pouvant être rencontrées...

Montage

Suivre les règles habituelles du montage des composants. Il n'y a pas de règle particulière de manipulation de composants à respecter. Tous les circuits sont en TTL.

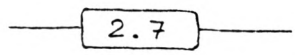
Faire attention aux points suivants :

1 - L'orientation des circuits intégrés n'est pas toujours identique,

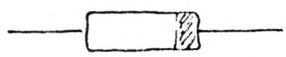
2 - Les diodes d1 et d2 sont identiques, la diode Z1 est une Zener et ressemble beaucoup aux précédentes. Ne pas la confondre.

3 - Les conducteurs "goutte tantale" sont polarisés. Il faut donc faire très attention, en les soudant, de ne pas se tromper de sens.

Les condensateurs "disque" ne sont pas polarisés.



La diode zener a un marquage 2.7 qui la différencie des deux autres diodes



Diodes d1 et d2

Liaison boucle de courant

Strap S3 : relier A à B

Liaison V 24

Strap S3 : relier A à C

Ne pas monter pour ce type de liaison le transistor T1 ni la diode zener Z1.

Pour les softwares MBC :
- Adresse 10 pour liaison périphérique -
- Adresse 11 pour liaison modem -

Adresse

Strap S1 : spécifie les unités de 0 à 9. Relier A au plot voulu.

Strap S2 : spécifie les dizaines 0 à 1. Relier A au plot désiré.

Adresse à utiliser impérativement pour le software standard (ASSEMBLEUR - BASIC...) : 10
— Sur S1 relier A à 0,
— Sur S2 relier A à 1.

Connexion au périphérique

Boucle de courant TELETYPE Modèle 33 ou 35

C'est le mode de liaison couramment utilisé sur les TELETYPES non modifiées :
— boucle de courant 20 mA,
— transmission full duplex.

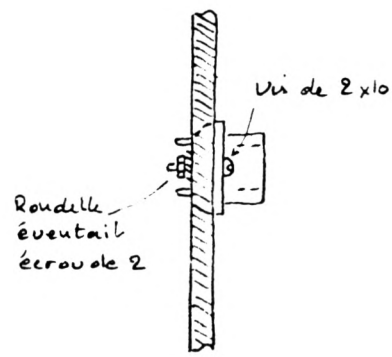
Pour accéder à la barette de connexion de la télétype, il faut enlever le capot plastique qui recouvre, en une seule pièce, l'ensemble clavier - imprimante.

Liaison V24 pour périphérique

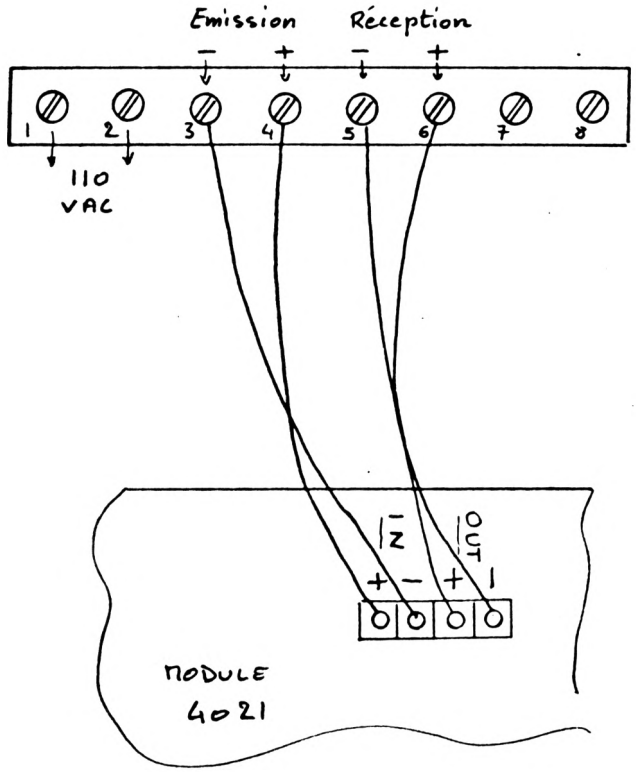
En premier, regardez les picots utilisés sur le connecteur de votre périphérique. Puis comparez-les à ceux qui sont marqués sur le schéma B.

Eventuellement, vous constaterez quelques différences.

fixation du connecteur femelle sur face arrière



Barrette de connexion TTY modèle 33 ou 35



Certains périphériques n'utilisent pas les lignes correspondant aux picots 5 et 6, d'autres utilisent la ligne 108 (picot 20).

Tout cela n'est généralement pas très grave.

Conformez-vous aux liaisons recommandées sur le schéma A : LIAISON PERIPHERIQUE.

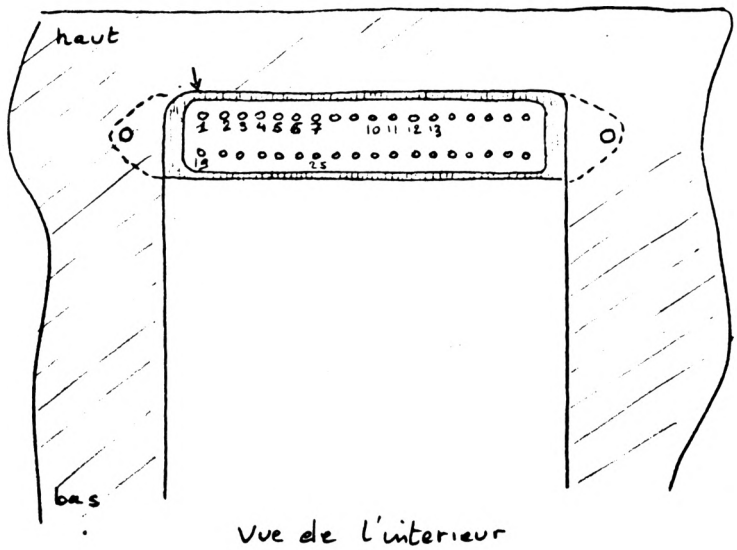
Connecteur face arrière ALCYANE

Si vous ne possédez pas de clavier MBC, mettez le connecteur en première position (en haut).

Pour liaison boucle de courant : picots 10 à 13.

Pour liaison V24 : picots 1 à 7 pour périphérique, (19 à 25 pour modem).

Nous vous conseillons de faire les essais avant de couper le câble plat de liaison.



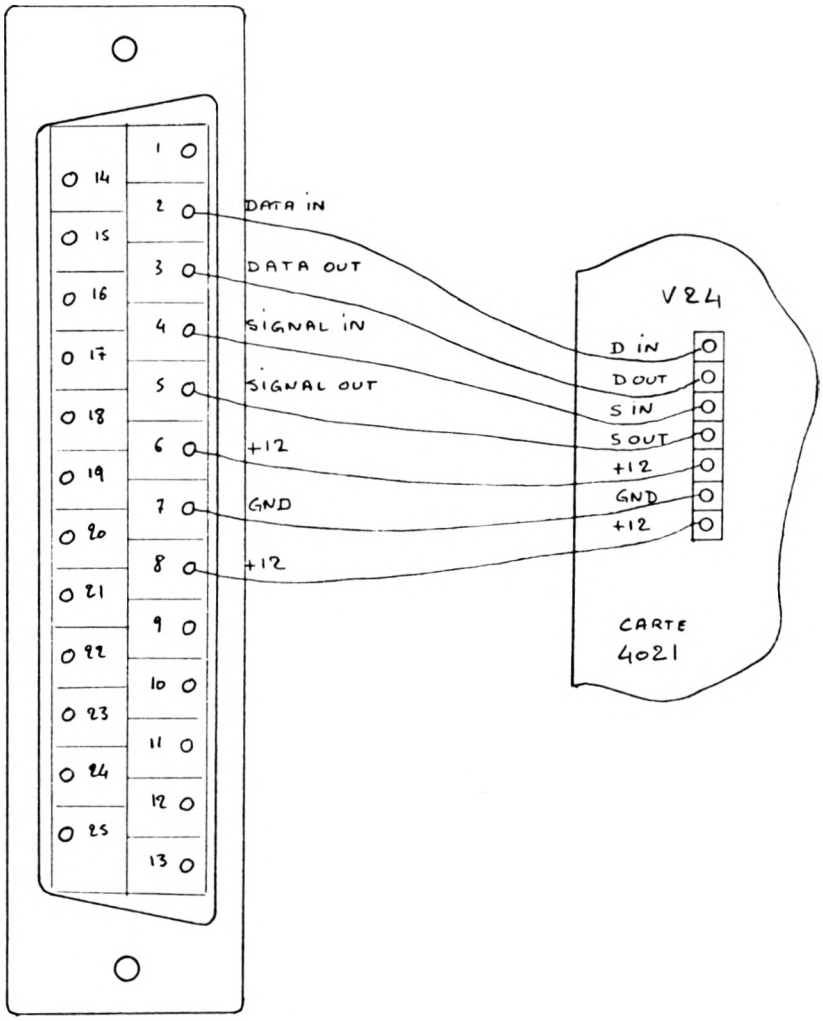
Vue de l'interieur d'ALCYANE

Picots

10	- OUT
11	+ OUT
12	- IN
13	+ IN
1	D IN
2	D OUT
3	S IN
4	S OUT
5	+12
6	GND
7	+12

LIAISON ALCYANE - PÉRIPHÉRIQUE

MBC ALCYANE



CONNECTEUR V24

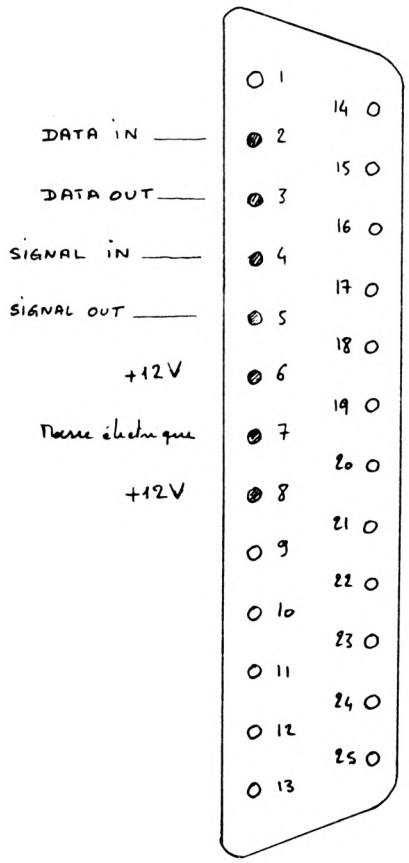
CÔTÉ PICOTS A SOUDER

SCHÉMA A

LIAISON ALCYANE - PÉRIPHÉRIQUE

MBC ALCYANE

SIGNAUX V24 GÉRÉS PAR ALCYANE MODULE 4021



Dénomination des lignes CCITT

1 - masse mécanique
2 - Emission de données
3 - Réception de données
4 - Demande pour emettre
5 - Prêt à emettre
6 - Modem prêt
7 - masse électrique
8 - Détecteur de porteuse

NOTA
ligne 1 non connectée
4 et 5 non gerées en standard
6 et 8 permanentes à 12 V

Liaisons prises en compte par la carte 4021

SCHÉMA B

LISTE DES COMPOSANTS 4021.

Circuits intégrés

D1 — 74 LS 42
D2 — 74 LS 00
D3 — 74 10
D4 — 74 LS 02
D7 — 74 12 6
D6 — 74 LS 74
D7 — LM 339

Transistor

T1 — 2N 2907

Diodes

d1 1N4148
d2 1N4148
Z1 (2,7v)

Capacité

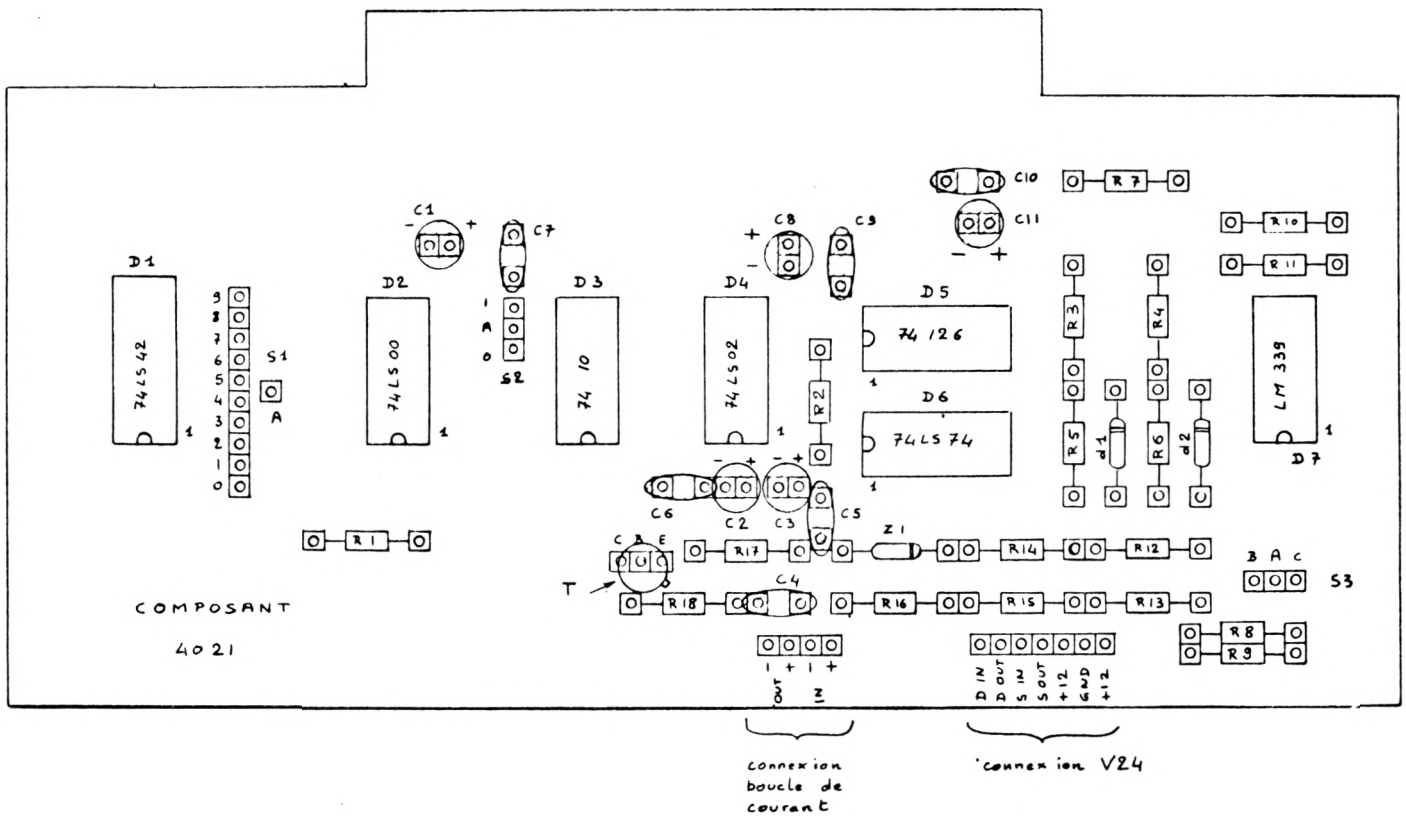
C1 — 2,2 µf goutte tantale
C2 — 2,2 µf goutte tantale
C3 — 2,2 µf goutte tantale
C4 — 100 nf
C5 — 100 uf
C6 — 100 uf
C7 — 100 uf
C8 — 2,2 µf goutte tantale
C9 — 100 nf
C10 — 100 uf
C11 — 2,2 µf goutte tantale

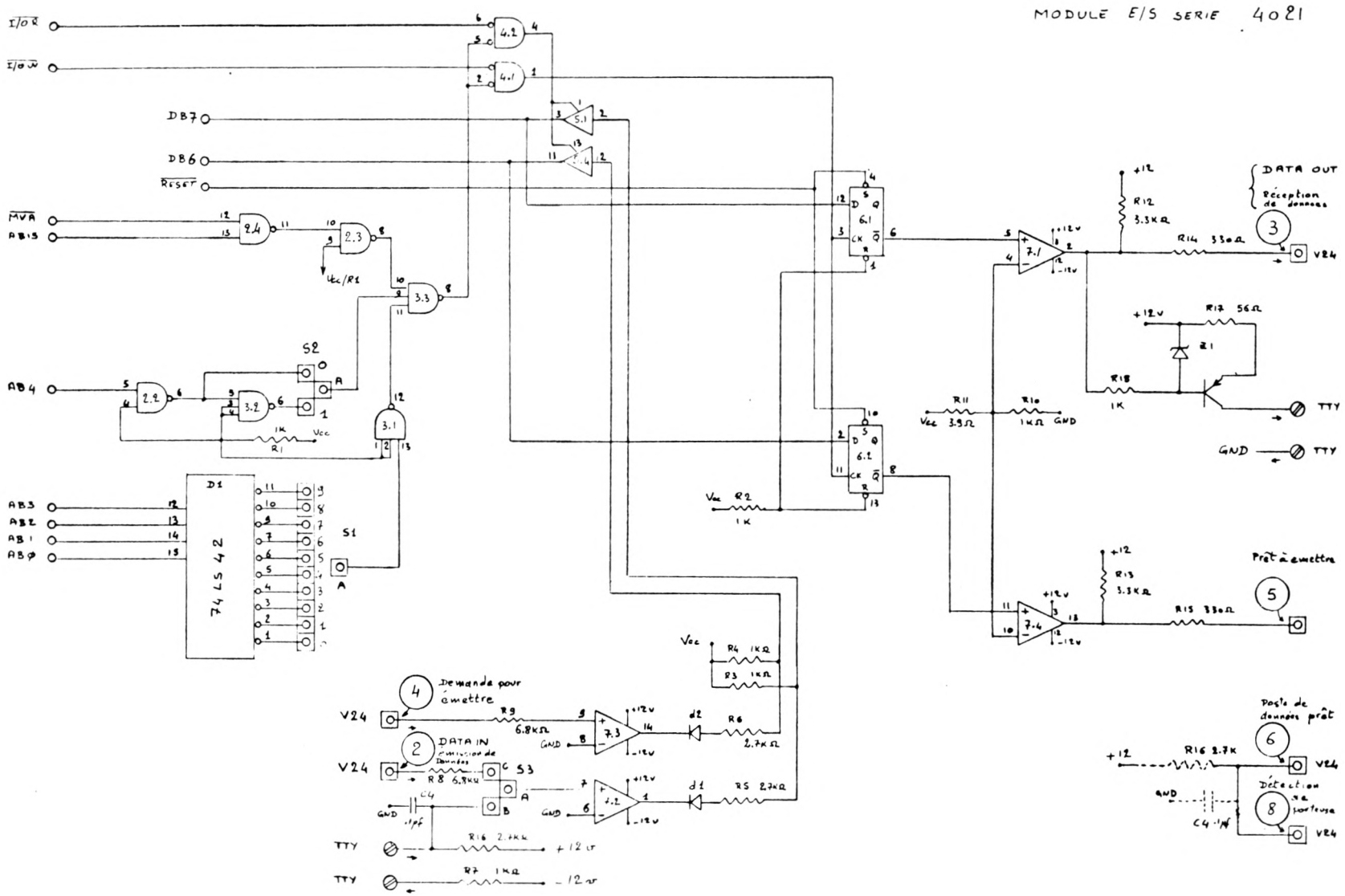
Capacité disque
100 nf non
polarisée

Resistances

R1 — 1 K
R2 — 1 K
R3 — 1 K
R4 — 1 K
R5 — 2.7 K
R6 — 2.7 K
R7 — 1 K
R8 — 6,8 K
R9 — 6,8 K
R10 — 1 K
R11 — 3,9 K
R12 — 3,3 K
R13 — 3,3 K
R14 — 330 Ω
R15 — 330 Ω
R16 — 2,7 K
R17 — 56 Ω
R18 — 1 K

MODULE E/S SERIE 4021




MODULE E/S SERIE 4021
I/OR
I/OW
DB7
DB6
RESET
MVA
AB15
AB4
AB3
AB2
AB1
AB0
74 LS 42
D1
S1
S2
S3
Vcc/R1
R1 1K
Vcc
R2 1K
R3 1K
R4 1K
R5 2.7K
R6 2.7K
R7 1K
R8 6.8K
R9 6.8K
R10 1K
R11 3.9
R12 3.3K
R13 3.3K
R14 330
R15 330
R16 2.7K
R17 56
R18 1K
Z1
DATA OUT
Réception de données
Prêt à emettre
Poste de données prêt
Détection de porteuse
Demande pour émettre
DATA IN
Émission de Données
V24
TTY
GND
+12
-12
+12V
-12V
C4 .1pf

7 - 9 - 4. Carte interface parallèle imprimante

Cette carte interface est composée de la carte universelle et d'une carte complémentaire d'adaptation pour imprimante. L'ensemble une fois monté est du format d'une carte standard Alcyane. Il permet d'interfacer les différents modèles d'imprimantes Mannesmann 80, 132 colonnes et 184 colonnes double chariot ; avec option minuscules/majuscules.

Principe du montage :

1 - réunion des deux demi-cartes,
2 - câblage du câble plat sur carte et connecteur imprimante,
3 - essai de fonctionnement,
4 - câblage du connecteur face arrière Alcyane.

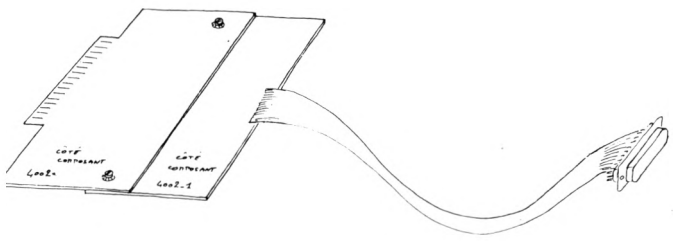
Attention : lors de la mise sous tension test de l'imprimante, ne jamais mettre en court-circuit avec un autre plot, le plot 17 (- 12 volts) du connecteur de l'imprimante.

Montage des cartes

Avant de souder les composants, vissez les deux cartes, 4002-1 sous la carte 4002, puis soudez les passants correspondant à la carte 4002-1 soit :

PBØ à PB7 sur FC4
PCØ à PC3 sur FC3
PAØ à PC4 sur FC2
2 plots GND sur FC1
2 plots + 5 sur FC1

Montez et soudez les composants de la carte au format standard ainsi réalisé.



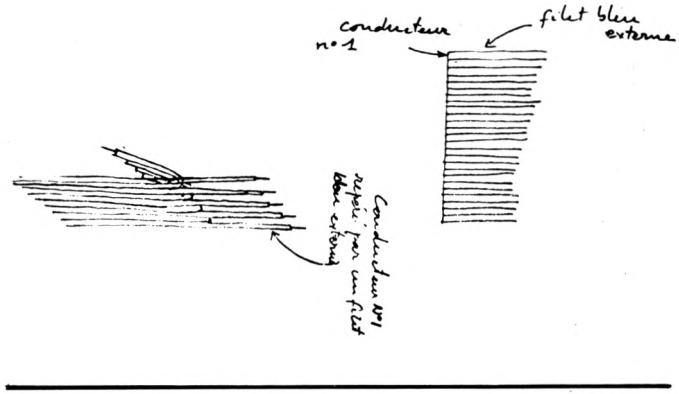
Il faut souder sur le dessus de la carte 4002 les pattes de la résistance R1.

Attention au double marquage D1, D2, D3, D4 des cartes 4002 et 4002-1, et ne pas confondre les DIL à y monter.

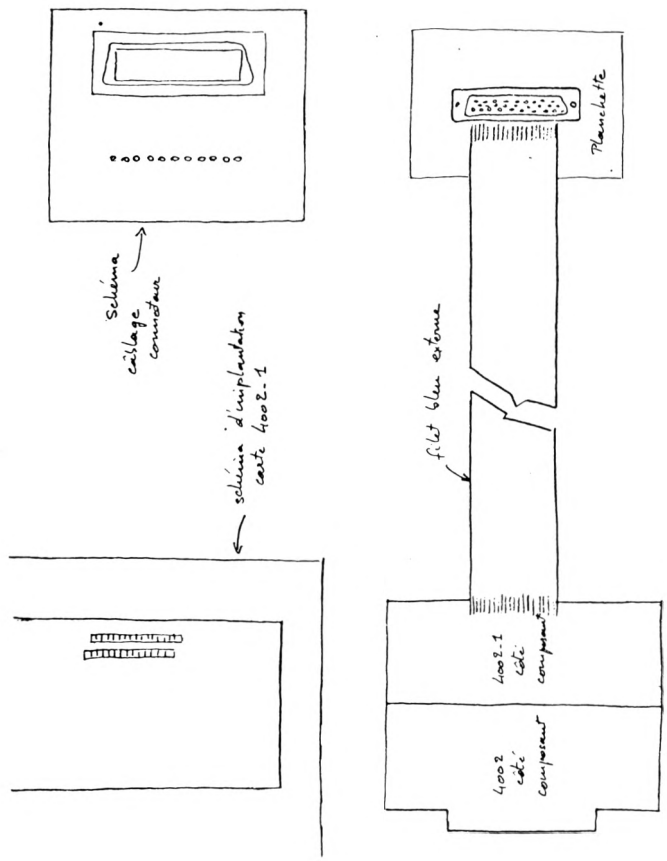
Les DIL de la carte 4002-1 ne sont repérés que sur le schéma d'implantation.

Liaison câble plat P80

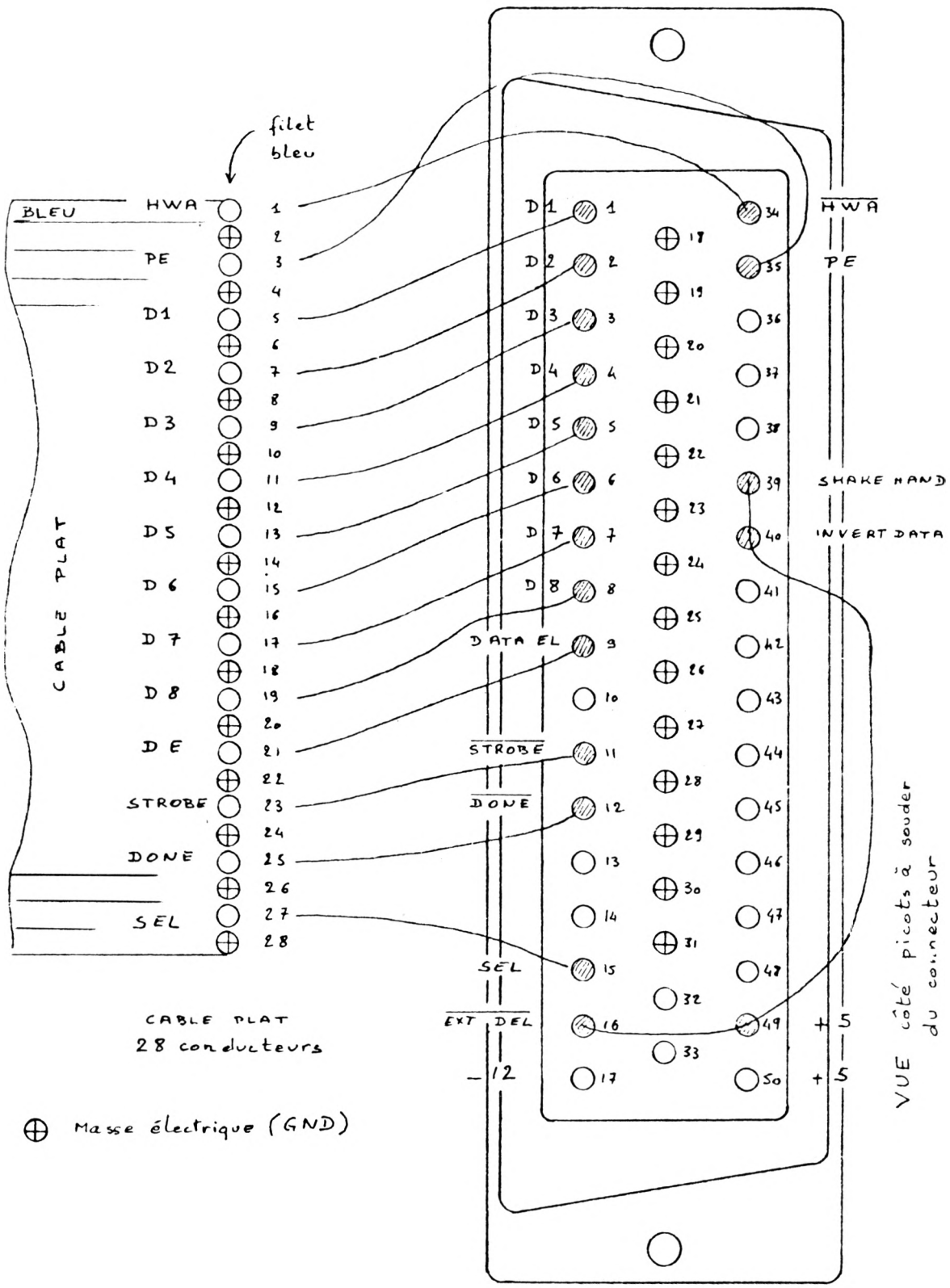
Pour l'imprimante Mannesmann P80 (80 colonnes), prendre un câble plat de 28 conducteurs. Les plots de 29 à 34 sont réservés aux imprimantes P132 et P184.



Disposition des éléments pour câblage du câble plat

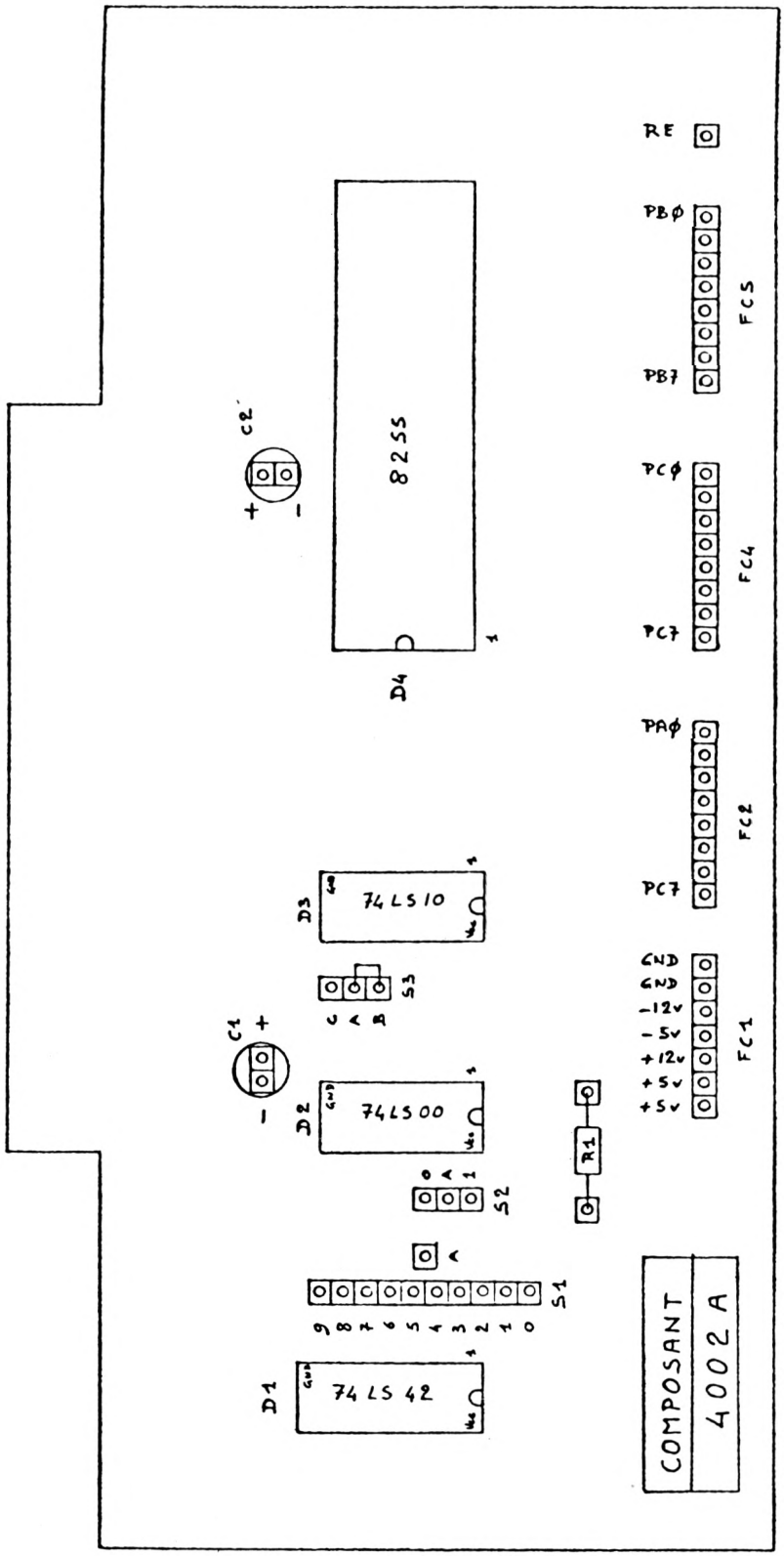


CABLAGE CONNECTEUR P 80



CARTE INTERFACE PARALLELE UNIVERSEL PROGRAMMABLE

4002 A



C1 tantale goutte 2,2µf
C2 " " "

R1 1 KΩ

S1 unité de l'adresse (BCD)

S2 dizaine de l'adresse (BCD)

(relier le plot A au plot correspondant à l'adresse voulue)

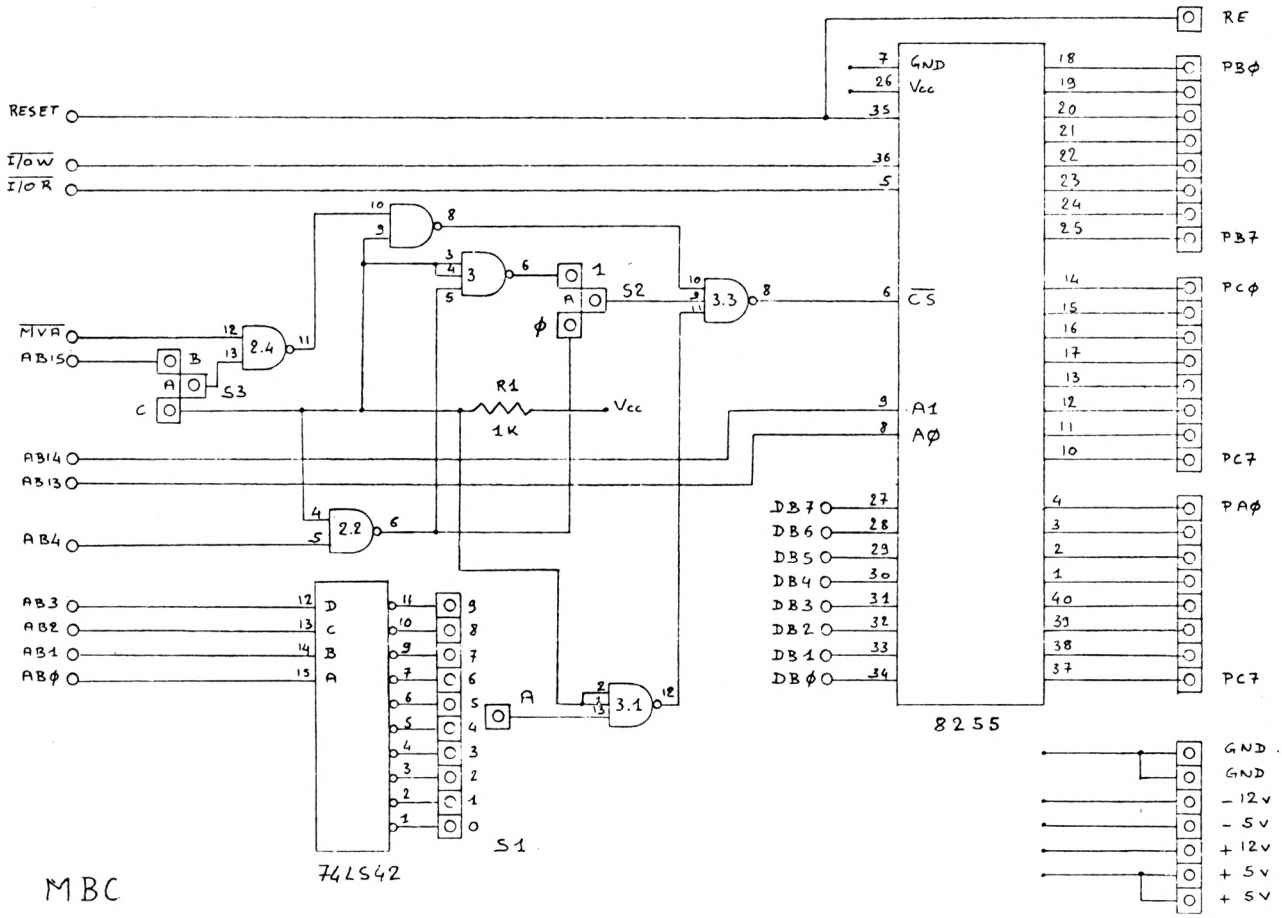
S3 relier A à B

ADRESSE STANDARD
Pour software MBC

Lecteur de ruban perfo. : 3
Imprimante 80/132/174 : 4
programmeur RPROM : 9
liaison Unité centrale à Unité centrale : 19

CARTE INTERFACE UNIVERSEL
PROGRAMMABLE

4002 A



MBC

P 80

CARTE ADAPTATION INTERFACE UNIVERSEL . 4002

POUR IMPRIMANTE MANNESMANN

4002 . 1

MBC

1ère Rangée du connecteur

coté câble	côté connecteur
5	1
7	2
9	3
11	4
13	5
15	6
17	7
19	8
21	9
23	11
25	12
27	15

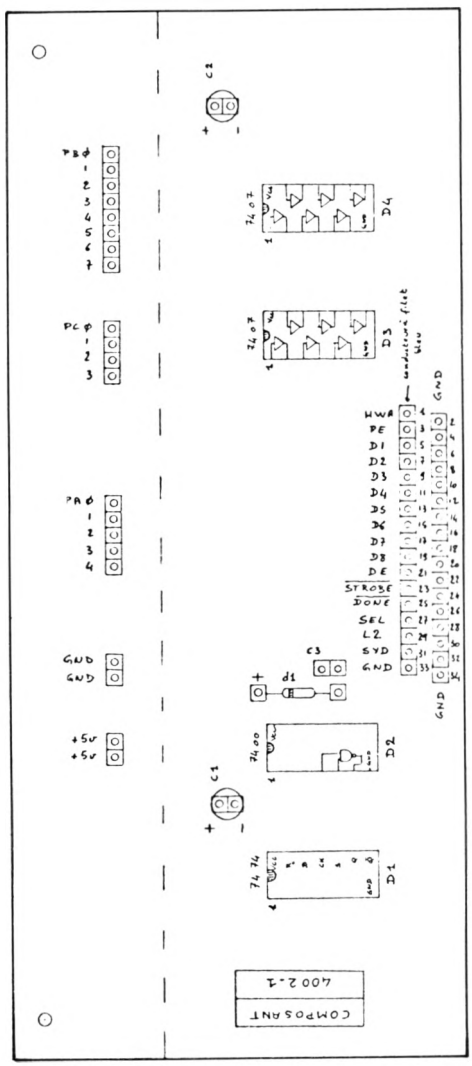
2ème Rangée du connecteur

2	18
4	19
6	20
8	21
10	22
12	23
14	24
16	25
18	26
20	27
22	28
24	29
26	30
28	31

3e Rangée du connecteur

1	34
3	35

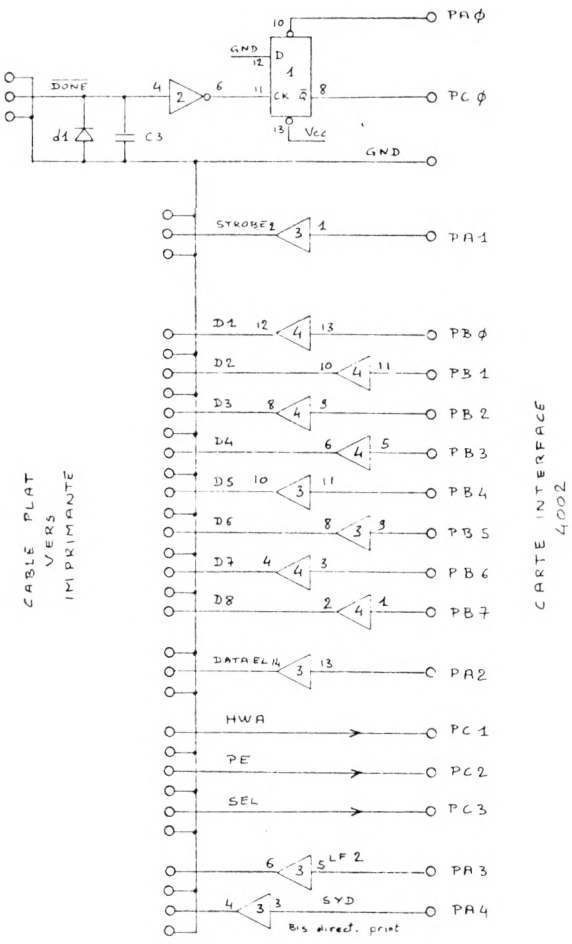
Puis sur le connecteur, relier par fil gainé
16 - 49 - 40 - 39

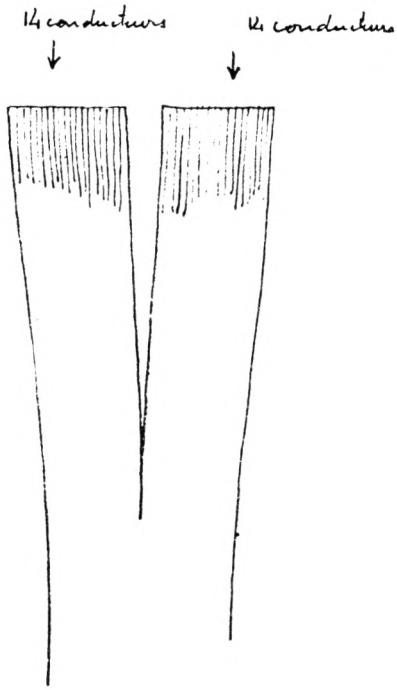


C1 2,2 µf tantale
C2 "
C3 1 nf
d1 Diode

4002_1 ADAPTATION IMPRIMANTE HANNESMANN
POUR INTERFACE 4002

MBC
ALCYANE





Aux deux extrêmités, séparer les conducteurs sur 4 cm, puis dénuder et étamer.

Pour faciliter le soudage du connecteur 50 contacts, fixez-le sur une planchette de bois.

Disposez à plat en face de vous la carte 4002/4002-1, le connecteur sur sa planchette, et les schémas du branchement dans le bon sens (celui du repérage).

Le câble plat doit aller, à plat, de l'un à l'autre.

Soudure côté carte du câble plat : pour faciliter le repérage, rabattre un fil sur deux en commençant par le conducteur numéro 2.

Tous les fils qui restent droits sont à souder sur les plots impairs, en commençant par le conducteur à filet bleu (externe) dans HWA, et ainsi de suite.

Souder ensuite tous les conducteurs restants, précédemment repliés dans les plots pairs (masse électrique GND).

Soudure côté connecteur du câble plat : rabattre de la même façon les conducteurs GND en commençant par le numéro 2.

Etamer les picots du connecteur, mais seulement ceux qui sont à souder pour auto-vérifier le câblage par la suite.

Souder dans l'ordre selon le tableau.

Câblage connecteur face arrière

Ne doit être fait qu'après les essais de bon fonctionnement de l'imprimante.

Retirez la carte interface du coffret, et coupez le câble à 50 cm du bord de la carte.

Fendre en son milieu, sur 10 cm, le câble plat, puis séparer les brins sur 4 cm, dénuder et étamer.

Etamez les picots utilisés du connecteur, rangée du dessus de 1 à 14, rangée du dessous de 19 à 32.

Soudez en premier le connecteur femelle.

La figure 3 donne le sens des éléments avec agrandissement du câblage du connecteur en figure 2.

Enfilez les capots du connecteur imprimante et du connecteur face arrière, puis soudez le connecteur mâle en faisant attention au sens de montage.

Montage définitif de la carte

Pour les softwares standards qui vous sont fournis, la carte interface imprimante doit avoir l'adresse n° 04.

strap S1 : A relié à 4
strap S2 : A relié à 0.

La carte doit être enfichée dans le connecteur spécifié par la feuille d'implantation des cartes qui vous est donnée avec Alcyane.

Faire passer le câble sous la tôle fond de panier. C'est fini...

CHAPITRE 8
PROBLEMES ACTUELS DE CONCEPTION

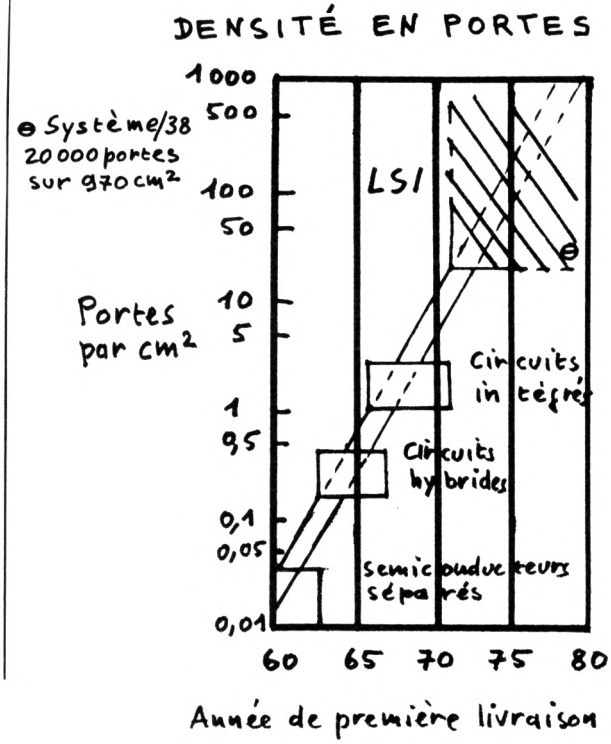
Notre propos va maintenant être d'essayer d'analyser comment une révolution technologique que nous commençons à connaître un peu mieux, va influencer très profondément l'utilisation de l'informatique et le marché des ordinateurs tel qu'il est aujourd'hui.

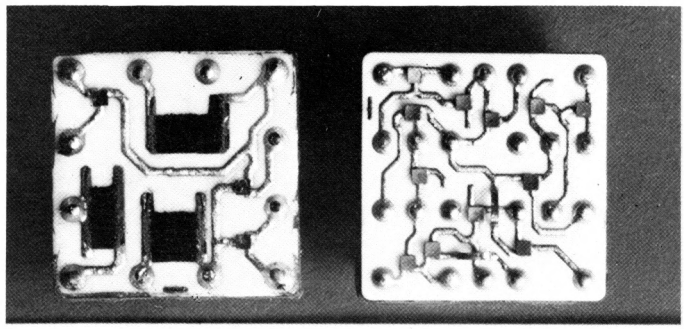
On peut dire que c'est un véritable bouleversement qui se prépare et que nos carrières, nos façons d'agir, vont s'en trouver progressivement de plus en changées, d'une manière que nous pensons d'ailleurs être bénéfique. Mais ce sera à vous d'en juger.

8 - 1. SMALL IS BEAUTIFUL

Commençons par jeter un coup d'œil au premier des nombreux graphiques qui vont illustrer ce chapitre. Comme nous l'avons vu en détail, une unité centrale quelconque d'ordinateur est composée essentiellement de quelques milliers de circuits logiques, les portes, qui concrétisent la structure logique, donc en fait les registres et les instructions qui ont été désirés par le concepteur de l'ordinateur. De nombreuses technologies peuvent être utilisées pour réaliser cette logique ; bien entendu nous ne nous attacherons ici qu'à la technologie électronique. Le graphique représente donc l'évolution de ladite technologie, au cours des vingt dernières années, en nous projetant d'ailleurs un peu dans l'avenir. En ordonnée nous trouvons le nombre de portes par cm^2. En fait ici ce nombre tient compte non pas de la densité absolue qui est obtenue sur les semi-conducteurs eux-mêmes, mais de la densité d'un ensemble complet de logique, y compris les interconnections entre les éléments, tel qu'on peut le trouver dans une unité centrale tout à fait classique. En abscisse nous trouvons l'année de première livraison, celle qui correspond par conséquent à l'apparition commerciale d'une génération bien définie d'ordinateurs.

On voit qu'en gros quatre technologies figurent dans ce diagramme. On part des semi-conducteurs séparés, qui sont caractéristiques des machines de seconde génération, que beaucoup d'entre nous connaissent et ont utilisés, des machines





SLT SLD

La technologie hybride SLT (Solid Logic Technology) correspond à l'apparition de la série 360. Il y a trois transistors et trois résistances (on voit les échancrures d'ajustage) sur le micromodule photographié ici. On remarque la petitesse des puces des transistors (1/2 mm, car les clichés sont ici grossis trois fois). Quelques années plus tard, la SLD a fait son apparition (Solid Logic High Density). Il y a cette fois une douzaine de transistors sur le carré de céramique. Où sont passées les résistances ? Sous le micromodule. Une dizaine de broches métalliques serties d'interconnexion ont été rajoutées. Dans la phase d'intégration suivante (non représentée ici), des puces de circuits intégrés SSI ont remplacé les transistors.

comme le 1401, la Gamma 30, le 7070 et toute la série des 7000 IBM, etc.

Déjà le progrès à ce niveau était très important et dû non seulement à la réduction de taille des semi-conducteurs, les transistors et les diodes, par rapport aux tubes à vide, mais surtout au fait qu'était apparu le circuit imprimé qui a permis de réaliser le câblage, non seulement de façon plus automatique, mais également dans un volume plus réduit. Cependant, nous constatons qu'il est nécessaire d'utiliser de 100 à 30 cm² pour une porte dans cette technologie de seconde génération.

8 - 1 - 1. La technologie hybride : SLT, SLD chez IBM

L'apparition des circuits hybrides marque un autre saut. Rappelons au passage qu'une seule série d'ordinateurs dans le monde a utilisé les circuits hybrides, et à première vue on pourrait se demander pourquoi nous en parlons ici : eh bien n'oublions pas que la série en question c'est tout simplement la série IBM 360, première série de machines à avoir été qualifiée de machines de troisième génération.

Dans les circuits hybrides c'est surtout une révolution d'emballage, si l'on peut dire, qui s'est produite, puisque les micro-plaquettes, c'est-à-dire les puces de silicium correspondant aux transistors (ici nous n'en sommes en effet pas encore aux circuits intégrés) contiennent chacune simplement un transistor ou une diode. Elles sont soudées sur un support de céramique de 11 mm de côté, à raison de trois ou quatre micro-plaquettes par support, l'ensemble de l'élément céramique recevant un capot métallique qui couvre toutes les micro-plaquettes d'un seul coup. La réduction, par conséquent, du brochage et des connections extérieures, a permis de gagner un facteur dix, en gros, par rapport à la technologie précédente, ce qui est tout à fait considérable.

Maintenant, pour réaliser une porte il nous faut environ 5 cm² en moyenne, ce qui est un gain de place important ; par contre, sur le plan du fonctionnement, nous faisons toujours appel aux mêmes semi-conducteurs, et rien n'est changé si ce n'est que des composants discrets tels que les résistances sont maintenant remplacés par des dépositions de film épais sur la céramique, l'ajustage de ces résistances étant effectué par une machine à sabler automatique qui échancre les films, d'où de très bonnes tolérances (2 à 3 %).

8 - 1 - 2. Les circuits intégrés

Mais c'est avec l'apparition des circuits intégrés que les gains deviennent encore plus importants : nous sommes maintenant à plusieurs portes par cm². C'est la technologie SSI (small scale integration, intégration à petite échelle, quelques portes, 2 à 4, par boitier) qui apparaît, et avec elle en particulier toute la fameuse famille logique

TTL dont nous connaissons déjà les caractéristiques électroniques pour les avoir étudiées en détail. A l'heure actuelle la TTL reste la famille la plus utilisée dans la conception des ordinateurs, nous y reviendrons tout à l'heure.

Lorsqu'on arrive au sommet du diagramme, en grimpant par conséquent à l'époque actuelle et en se projetant très légèrement dans le futur, on est dans la LSI (large scale integration, intégration à grande échelle), avec des densités en portes qui paraissent pharamineuses, par rapport à tout ce qu'on a vu jusqu'ici. On en est maintenant aux dizaines, aux centaines et même aux milliers de portes par cm^2.

Alors, à la vue de ce diagramme, la réaction ne se fait pas attendre. Les progrès de la micro-électronique qu'il retrace sont tels que l'on se demande bien pourquoi dès aujourd'hui un 370/168 ne tient pas sur un timbre-poste, et pourquoi il coûte encore aussi cher !

Il y a donc par conséquent quelque chose qui ne va pas, quelque chose qui n'est pas apparent sur ce diagramme. C'est ce que nous allons essayer d'expliquer maintenant.

8 - 1 - 3. Problèmes de conception des ordinateurs lourds : le mur du son

La confusion

En effet, ce graphique ne signifie pas grand-chose sur le plan pratique, et ceci pour deux raisons essentielles : d'une part il mélange logique et mémoire, et d'autre part il ne tient pas compte des problèmes de conception des ordinateurs.

Commençons par le premier point. La fonction de mémorisation en informatique est quelque chose de parfaitement défini. Il s'agit de stocker des bits de façon temporaire dans la mémoire centrale de l'ordinateur, et d'ailleurs également dans ses registres.

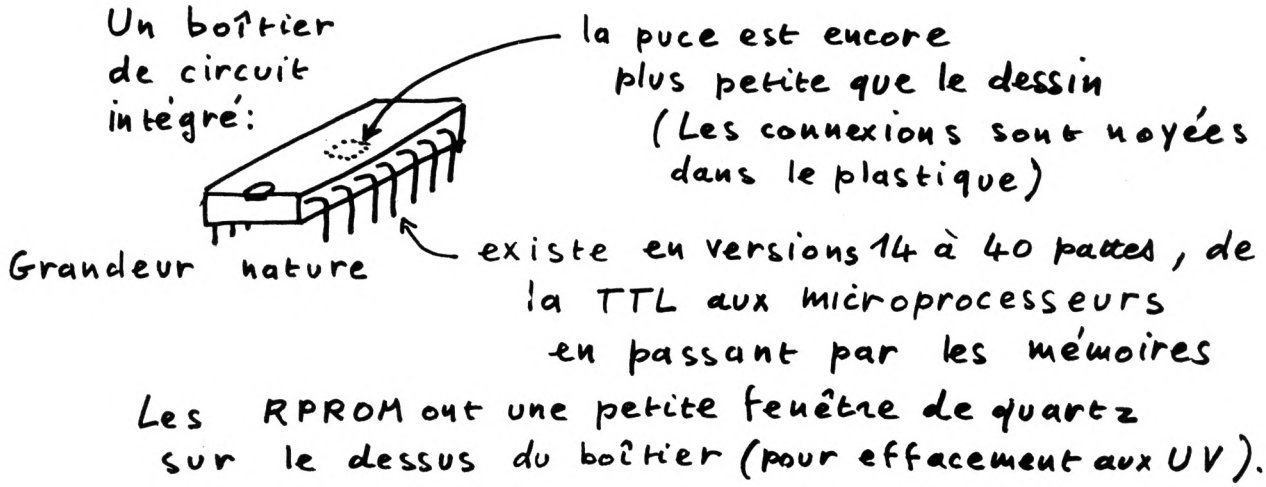
Ici, en ce qui concerne le stockage d'informations, les progrès de la micro-électronique ont effectivement été spectaculaires.

En effet, dans une mémoire, cette fonction de stockage constituant une nécessité très clairement définie, on y répond par le dessin de circuits intégrés qui ressemblent à de véritables festons, des motifs répétitifs, rangés par milliers les uns à côté des autres, représentant les cellules de stockage binaire.

Nous avons vu que diverses technologies peuvent être utilisées, en particulier stockage dans des cellules statiques ou des cellules dynamiques, nous n'y reviendrons pas. Bornons-nous simplement à dire que cette fonction de mémorisation est une fonction bien reconnue, nécessaire dans tout ordinateur, et qu'il a donc été facile de la standardiser.

Qu'a-t-on fait ? Eh bien on a créé tout simplement des circuits intégrés de capacité normalisée, croissant au fur et à mesure que la technologie progressait. Ils ont été capables de stocker 128 bits, puis 256, puis 512, puis 1024, etc., ces choix binaires ronds correspondant à une utilisation maximale des lignes d'adressage, toujours en binaire.

Le plafond disponible à l'heure actuelle (1978) est de 4096 bits en technologie statique, et de 16384 bits en technologie dynamique. Notons cependant que ces hautes densités commencent seulement d'apparaître sur le marché, et pour l'instant restent relativement chères, rapportées aux bits stockés. Les coûts les plus abordables, sont au niveau des circuits 1024 bits statiques et 4096 dynamiques. De plus, et cette constatation resservira dans la suite, ce haut degré d'intégration, qui sera un facteur d'économie à moyen terme, n'existe pas lorsque les mémoires doivent être d'un accès très rapide. Ceci sera un facteur



d'augmentation de coût important pour les machines rapides, par rapport aux machines moyennes, ce qui sera déjà un désavantage pour le gros ordinateur.

Si pour ce dernier nous essayons au contraire de profiter d'une économie maximale sur le prix des circuits et de faciliter la conception par l'utilisation de mémoire statique, ce qui entraîne également une amélioration de sécurité, ces trois points simultanés correspondent aux 1 K statiques. Or la consommation électrique des boîtiers statiques 1 K s'avèrera prohibitive pour des machines importantes dans lesquelles l'un des paramètres principaux est la dissipation de chaleur, comme nous le verrons plus loin, ce qui conduit au contraire à essayer d'utiliser des boîtiers fortement intégrés, donc assez chers, dont la consommation électrique, et par conséquent la dissipation calorifique décroissent au fur et à mesure que l'intégration augmente puisque la cellule unitaire de stockage devient de plus en plus petite.

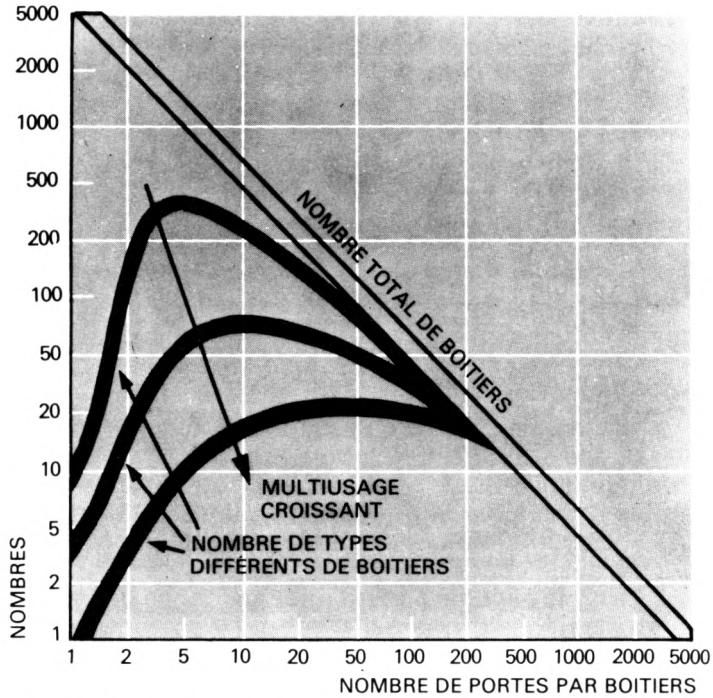
Ceci est donc le cas des boîtiers mémoire ; nous allons voir qu'il est très différent de ce qui se passe dans le cas de la logique.

En effet pour la logique le problème est beaucoup plus aigu. L'unité centrale de l'ordinateur, outre la mémoire, contient le processeur ; ce dernier est un écheveau de logique, d'apparence assez confuse, que l'on appelle d'ailleurs souvent logique aléatoire (random logic) constitué de milliers de portes logiques qui matérialisent les fonctions booléennes, et donc le répertoire d'instructions désiré.

Pour constituer ce processeur, nous allons utiliser une certaine quantité de boîtiers de circuits intégrés, contenant un certain nombre de portes, nombre qui dépend non seulement de la technologie des circuits intégrés au moment considéré, mais aussi d'un choix de standardisation, et c'est ici que le bât va commencer à blesser.

Regardons en effet le second diagramme. Il concerne un exemple précis que nous prenons comme illustration pour analyser les difficultés auxquelles sont confrontés les constructeurs d'ordinateurs, de taille moyenne ou importante, qui voudraient bien utiliser les possibilités offertes par l'intégration. Mettons-nous dans le rôle du groupe d'ingénieurs qui a à concevoir le processeur d'un ordinateur moyen comportant 5000 portes. Pour réaliser ce processeur nous allons faire un choix de boîtiers de circuits intégrés, mais lesquels allons-nous employer ?

Deux questions peuvent être posées : quel nombre total de boîtiers faudra-t-il pour construire le processeur, et de combien de types différents ? Ce sont ces deux paramètres qui sont portés en ordonnée sur le graphique. Ils correspondent bien entendu à deux tracés différents, la droite inclinée d'une part et la famille de courbes en cloche d'autre part.



Nombre total et nombre de types différents de boîtiers nécessaires pour fabriquer un processeur de 5000 portes.

Mettons-nous tout d'abord dans le cas de l'utilisation d'une intégration modeste, de la SSI, c'est-à-dire de la TTL tout à fait ordinaire, moins avancée même que la TTL actuelle, avec par exemple une seule porte "nand" par boîtier (porte NON ET). Pourquoi une porte nand ? Eh bien tout simplement parce qu'il est possible, comme nous l'avons vu, de constituer entrèrement un ordinateur en utilisant seulement des portes nand. Nous serons ici à la standardisation maximale, et nous utiliserons 5000 boîtiers pour fabriquer notre processeur, mais ces 5000 boîtiers seront d'un seul type, la porte nand. Nous sommes donc ici tout à fait à gauche de notre graphique. Dans la pratique, l'intégration disponible sera un peu plus forte et nous aurons par exemple quatre portes nand par boîtier.

Compte-tenu de la taille moyenne des circuits imprimés que l'on utilise et du nombre de boîtiers qu'ils supportent, nous pourrons dire que notre processeur, à l'exclusion bien sûr de la mémoire centrale, représentera une centaine de circuits imprimés différents. Ces circuits imprimés seront tous à concevoir, dessiner et interconnecter. Le travail de conception globale pour le processeur sera considérable. Nous trouverons donc, chez le constructeur, des bureaux d'étude comportant des centaines d'ingénieurs et représentant des investissements de développement élevés.

Sautons maintenant à l'autre extrême en nous projetant dans l'avenir. Supposons que l'intégration, plus poussée que la LSI que nous connais-

sons aujourd'hui, soit telle que *tout* processeur puisse tenir dans un seul boîtier. Il nous faudra alors un seul boîtier, et d'un seul type, pour faire le processeur. Muni de ses éléments annexes il tiendra sur un seul circuit imprimé. Il est évident que si nous supposons que ce boîtier est commercialement disponible, le travail de conception de l'ordinateur sera considérablement réduit.

Est-ce à dire que le grand bureau d'étude de conception dont nous parlions tout-à-l'heure aura disparu ? Non, bien sûr. Il existera toujours, mais il sera ailleurs, sans doute chez le fabricant de circuits intégrés, à moins que le constructeur n'ait un débit de matériel si important qu'il lui soit possible de justifier économiquement chez lui le développement du circuit en question. Une évolution est donc ici enclenchée, et nous ne pouvons que nous demander quel sera son terme.

Il est évident, à la limite, que l'on peut s'interroger sur ce que deviendra l'industrie informatique de pointe ; ne se fusionnera-t-elle pas complètement avec l'industrie électronique ? Ce qui représenterait une redistribution des forces et un bouleversement sans précédent sur le marché des ordinateurs.

Le mur du son

Mais laissons provisoirement cette question, et venons-en au cas actuel, qui est le cas intermédiaire. Nous sommes cette fois en MSI (medium scale integration), capables de mettre un certain nombre de portes par boîtier, peut-être 30 ou 50 portes sur un seul boîtier. La question se pose maintenant à nous : comment découper la logique de notre processeur ?

Si nous opérons sans détour et, permettons-nous une image, en prenant le dessin logique théorique du processeur et en le découpant avec une paire de ciseaux en une centaine de morceaux tous différents (ce qui représentera bien nos 5000 portes), il est clair que nous aurons à utiliser dans ce cas une centaine de boîtiers qui seront *tous* différents les uns des autres, et il n'y aura aucune raison pour que ces boîtiers électroniques puissent exister sur le marché. Chacun d'eux sera à concevoir spécialement.

Il est évident dans ce cas que cette opération n'est absolument pas justifiée économiquement. En effet tout le travail de conception logique devra d'abord être fait comme si notre ordinateur était fabriqué en TTL classique, puis il faudra ajouter au coût de cette conception celui de la conception physique des masques de circuits intégrés à fabriquer, et l'amortissement de cette conception ne sera possible que si nous avons à fabriquer un nombre d'ordinateurs très élevé, se chiffrant par dizaines ou par centaines de mille.

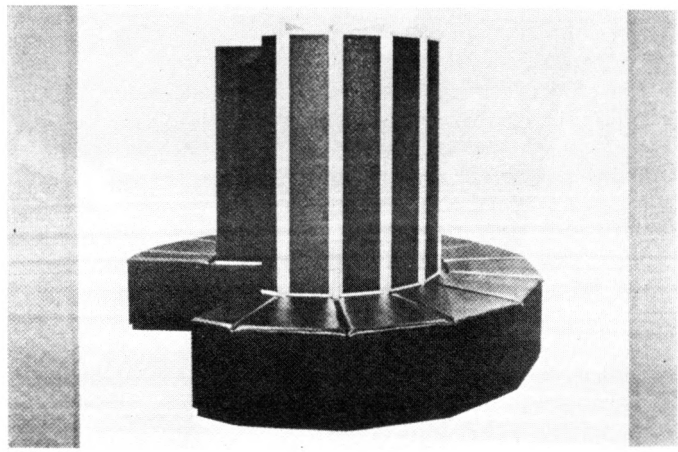
Le marché mondial des ordinateurs ayant le plus de succès est bien loin de ces chiffres à l'heure actuelle, surtout si nous tenons compte qu'il est quand même divisé entre un certain nombre de constructeurs, et qu'ici nous raisonnons au niveau d'un constructeur.

Si nous nous montrons très astucieux, nous réussirons au prix d'un effort de conception plus important à isoler des fonctions d'usage général répétitives à l'intérieur du réseau de logique aléatoire de notre processeur. Dans ce cas nous pourrons retrouver par-ci par-là des boîtiers identiques ou à peu près identiques, ce qui nous permettra de diminuer un peu le nombre de types de boîtiers à concevoir : ceci correspond à une recherche de standardisation menant à des boîtiers à usages multiples, figurée par les courbes en cloche de plus en plus basse sur le graphique. Cet effort supplémentaire, qui sera lui-même coûteux, allègera un peu notre fardeau, mais sans vraiment résoudre le problème. Le travail et le coût de conception restent très élevés et s'amortiront difficilement sur les séries moyennes de composants qui sont fabriquées. En bref, nous pourrions dire que nous sommes dans la même situation que l'aéronautique, au cours des années qui suivirent la deuxième guerre mondiale : nous n'avons toujours pas réussi à franchir le mur du son.

8 - 1 - 4. Gros, complexe et simple à la fois

Nous allons prendre un exemple précis pour étayer cette démonstration. Faisons une analyse rapide du Cray-1, le plus puissant ordinateur du monde, nommé d'après son père fameux, fameux pour les gros ordinateurs Control Data.

Après de nombreuses années passées comme directeur de la recherche de Contral Data, Seymour Cray a fondé une société dont le premier produit a été le Cray-1, qui bien sûr comme ses prédécesseurs a continué d'être le plus puissant du monde. Cette machine a un cycle mémoire de 50 nanosecondes, une horloge à 12,5 nanosecondes ; elle peut effectuer 80 millions d'opérations



flottantes par seconde. On peut dire que c'est le sommet de la pyramide des ordinateurs mondiaux.

Seymour Cray, est justement réputé comme l'un des plus géniaux concepteurs d'ordinateurs. Il n'est que de voir le succès de ses précédentes machines. Qu'est-ce que le Cray-1 emploie comme boîtier ? Quatre types de boîtiers seulement. Pour la mémoire, des boîtiers à 1024 bits, pour la logique, qui est la partie qui nous intéresse, des boîtiers à deux portes "nand" seulement, en un type rapide et un type lent, et des registres à 16 bits. Nous voyons que la logique du Cray-1 est *entièrement réalisée en SSI.* Il s'ensuit que le nombre des éléments est très élevé. La machine comprend plus de 1600 circuits imprimés à cinq couches, de 113 types différents, portant chacun plus d'une centaine de boîtiers, ce qui fait dans les 200 000 boîtiers au total. A une quinzaine de transistors équivalents par boîtier, il y aura deux millions et demi de transistors dans le processeur, plus 100 millions d'autres dans les mémoires pour une mémoire de taille moyenne : il va sans dire qu'il y a intérêt à bien tout refroidir...

Le travail de conception d'une telle machine est considérable. N'oubliez pas que nous avons parlé de circuits imprimés à cinq couches. Vous vous rappelez des étapes successives de conception logique d'implantation puis de réalisation des mylars des circuits imprimés ; le travail est déjà délicat pour des circuits imprimés double-face, vous pouvez facilement imaginer à quel niveau de complexité il monte pour des circuits imprimés comme ceux-ci. Nous voyons donc que cette machine se situe tout à fait à gauche du diagramme que nous venons d'étudier. Elle n'est absolument pas intégrée. Sa compacité, qui est visible sur la photographie, n'en est que d'autant plus remarquable. Sur le plan dimensions, le Cray-1 n'a que 1,24 m de diamètre et pèse environ cinq tonnes. Le dégagement de chaleur dans un volume aussi réduit est très important par unité de volume. La raison primordiale pour laquelle le volume doit être réduit est d'ailleurs que les interconnexions doivent rester les plus courtes possibles, à cause des temps de transit des signaux, qui commencent à devenir significatifs par rapport au temps de cycle pour des machines d'une rapidité aussi élevée : ils se propagent à peu près à la vitesse de 20 cm par nanoseconde.

Seymour Cray a déclaré, au cours d'une conférence, que les deux problèmes principaux posés par la conception des ordinateurs de très grande puissance étaient, d'une part d'arriver à se débarrasser convenablement de la chaleur, et d'autre part les difficultés causées par l'épaisseur du "mat".

Le deuxième cliché montre ce dont il s'agit. C'est l'épais écheveau d'interconnexions des circuits imprimés entre eux, ce qu'au niveau d'un ordinateur plus modeste nous appellerions le fond du panier, ou le bus pour les machines qui en ont un. La photographie donne une petite idée du problème pratique posé ; le refroidissement de la machine est entièrement effectué à l'aide d'une circulation interne de fréon qui enlève les calories aussi près que possible de leur source.

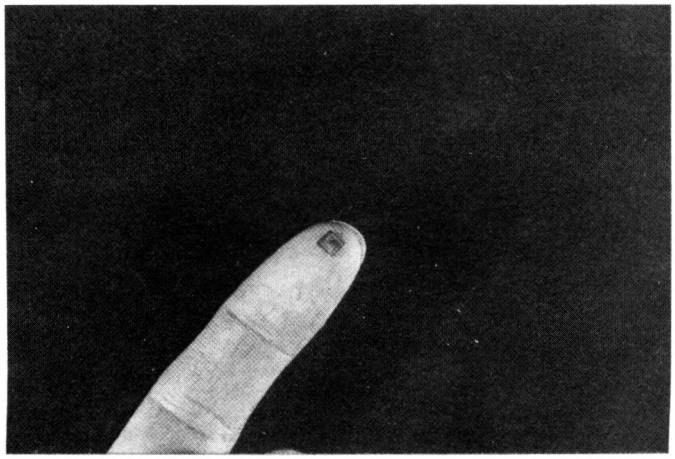
En conclusion, compte-tenu de l'état de la technologie et des problèmes de conception qui se posent, on peut dire qu'à l'heure actuelle un ordinateur de très grande puissance est obligatoirement fabriqué dans une technologie très peu intégrée, et par conséquent n'est absolument pas en état de profiter des avantages économiques qu'offre la LSI.

Question maintenant : existe-t-il des machines qui en ce moment sont déjà placées à droite du diagramme ? Eh bien, oui. Et ce sont évidemment les microprocesseurs.

8 - 2 SMALL IS BEAUTIFUL; AGAIN

Par principe même, en effet, les microprocesseurs sont des processeurs ayant la complexité maximale que l'on sait fabriquer en grande série sur une seule micro-plaquette de silicium, un chip de quatre millimètres de côté environ.

Devant la quasi impossibilité de standardiser des fonctions au niveau intermédiaire, comme nous l'avons vu tout-à-l'heure, voici donc la solution globale radicale : une standardisation au niveau de la fonction complète, le processeur entier. On obtient donc ainsi un boîtier qui va pouvoir être d'usage très général. Etant d'usage très général, il pourra donc être fabriqué par centaines de mille, vu la demande du marché. Ceci ramènera donc à un coût très bas les frais de développement du boîtier rapporté à l'unité, les frais de fabrication étant faibles. Le microprocesseur est donc une excellente mesure de ce que permet de faire la meilleure technologie disponible à un moment donné.



Une standardisation de fonctions radicale : le microprocesseur, qui a tranché le nœud gordien de l'indécision pour le choix des standardisations de modules logiques. Photo de la puce avant la mise en boîtier : elle contient dix mille transistors.

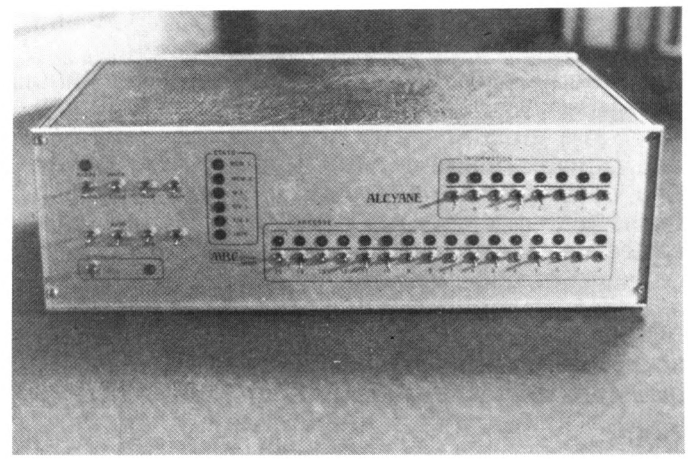
Rappelons rapidement, en raccourci, une évolution que nous avons par ailleurs analysée en détail. Le premier microprocesseur inventé par Intel est apparu en 1971. Il traitait des mots de 4 bits et possédait une quarantaine d'instructions. La seconde génération, dont on imaginait qu'elle puisse apparaître quelques années plus tard, est en fait arrivée l'année suivante : 8 bits, 75 instructions. Cette fois-ci, par conséquent, la possibilité de traiter l'information, octet par octer, l'octet étant véritablement l'unité informationnelle de base, aussi bien dans les ordinateurs importants que dans la transmission de données. Quant à la troisième génération, surprenant encore tous les observateurs, elle a vu le jour en 1974, grâce à la nouvelle technologie "MOS" canal "N" à grille de silicium, toujours à octets.

Cette troisième génération de microprocesseurs comportait plus d'une centaine d'instructions de programmes et permettait de dérouler les programmes dix à vingt fois plus rapidement que la seconde génération. Des instructions de longueur variable, 8, 16 ou 24 bits, un champ d'adressage non tronqué de 16 bits, évitant toute pagination dans un espace de 64Ko, une structure moderne avec quatre registres généraux de 16 bits, un pointeur de pile pour retour automatique de sous-programmes : voilà les caractéristiques essentielles du microordinateur typique de troisième génération. Bien entendu, tout seul, ce boîtier ne sert à rien : il doit être entouré de quelques centaines d'autres boîtiers de circuits intégrés pour constituer une unité centrale. Cependant, de par son existence même, il a ouvert une nouvelle ère de l'informatique : *il permet en effet à de très petites équipes de concevoir, mettre au point, fabriquer et commercialiser des ordinateurs. C'est un phénomène historiquement nouveau en informatique.*

La révolution technico-économique est donc double. D'une part, baisse du coût de la conception, d'autre part baisse du coût des éléments du hardware : elle se place donc essentiellement au niveau des coûts, et nous tâcherons de la quantifier chaque fois que possible.

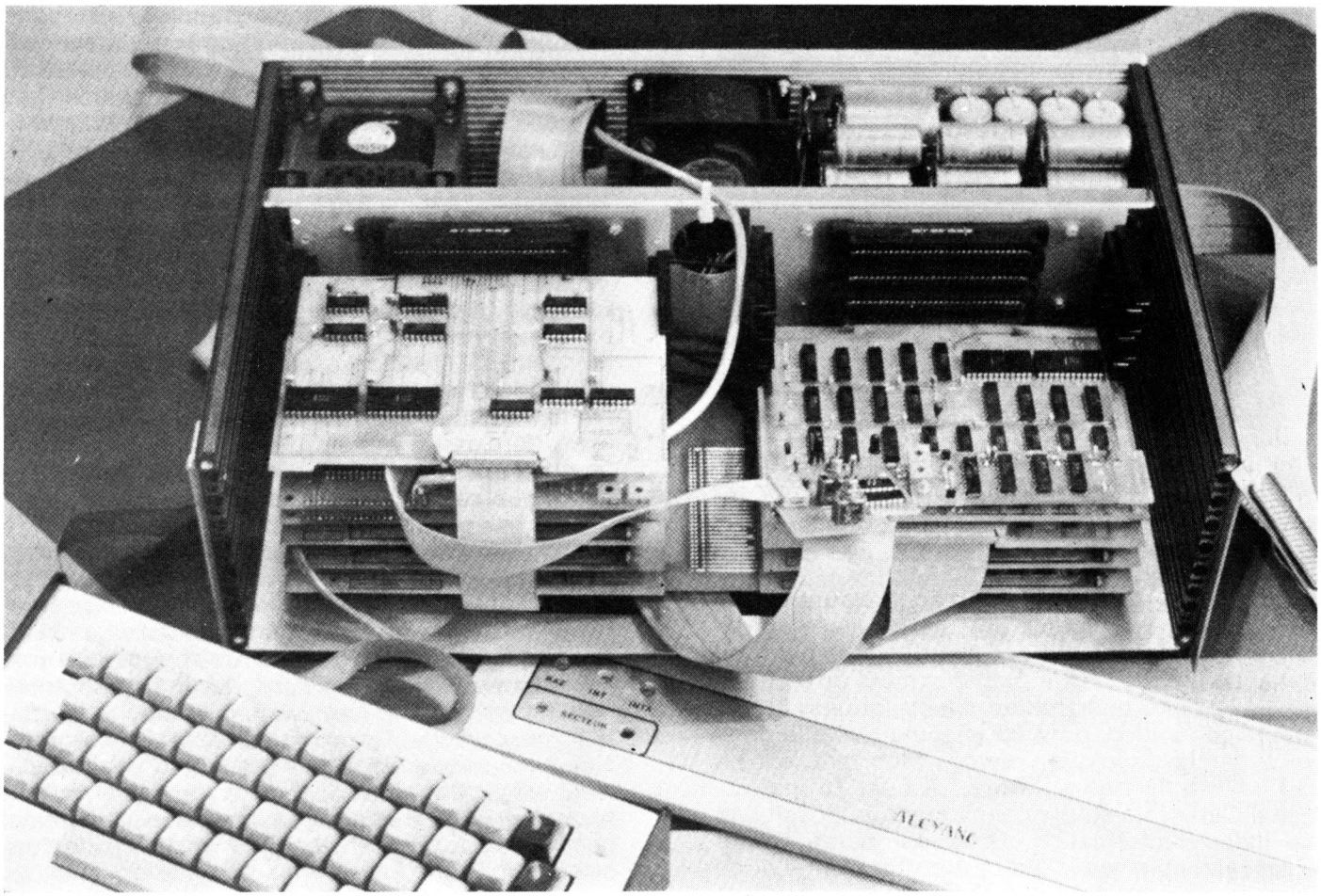
La photo suivante nous montre le produit complètement terminé, une unité centrale de microordinateur complète. La conception d'une unité centrale de ce type peut se réaliser de la façon la plus efficace, à notre avis, à l'aide de petits groupes polyvalents où, avec des ratios différents, tout le monde est compétent à la fois en hardware et en software. Notons à cet égard une différence importante par rapport au processus de spécialisation de plus en plus étroit auquel on assiste dans l'industrie depuis de nombreuses années, quelles que soient les branches. Il s'ensuit d'ailleurs que ce genre de profil devient actuellement de moins en moins répandu.

On peut compter qu'il faut trois bonnes années de travail acharné, pour arriver au but. Cette période est jalonnée par une succession d'étapes qui vont du design général, tant hardware que software, à la réalisation de plusieurs prototypes successifs, puis au passage progressif à l'industrialisation du produit. Celui-ci subit ensuite un debugging continu, et de nombreuses mises à niveau technique. En effet, il y a loin de l'ordinateur idéal, théoriqe, conçu sur le papier, à une machine robuste et exploitable en service continu.



L'unité de traitement : une unité centrale de seconde à troisième génération pour 99,5 % moins cher. Elle tient 99,5 % moins de place.

Citons parmi les difficultés rencontrées : tout d'abord la différence géométrique dans l'espace entre le prototype final et le produit fini industrialisé. Cette différence joue sur de nombreux points, aussi bien électrique que mécanique ou thermique, bien entendu. Par ailleurs, la propreté



L'unité de traitement ouverte : on voit les modules (à droite, sur le dessus, le module de visualisation). Noter la connectique en câbles plats. Le coffret peut contenir 14 modules, et une unité de traitement peut comporter deux coffrets.

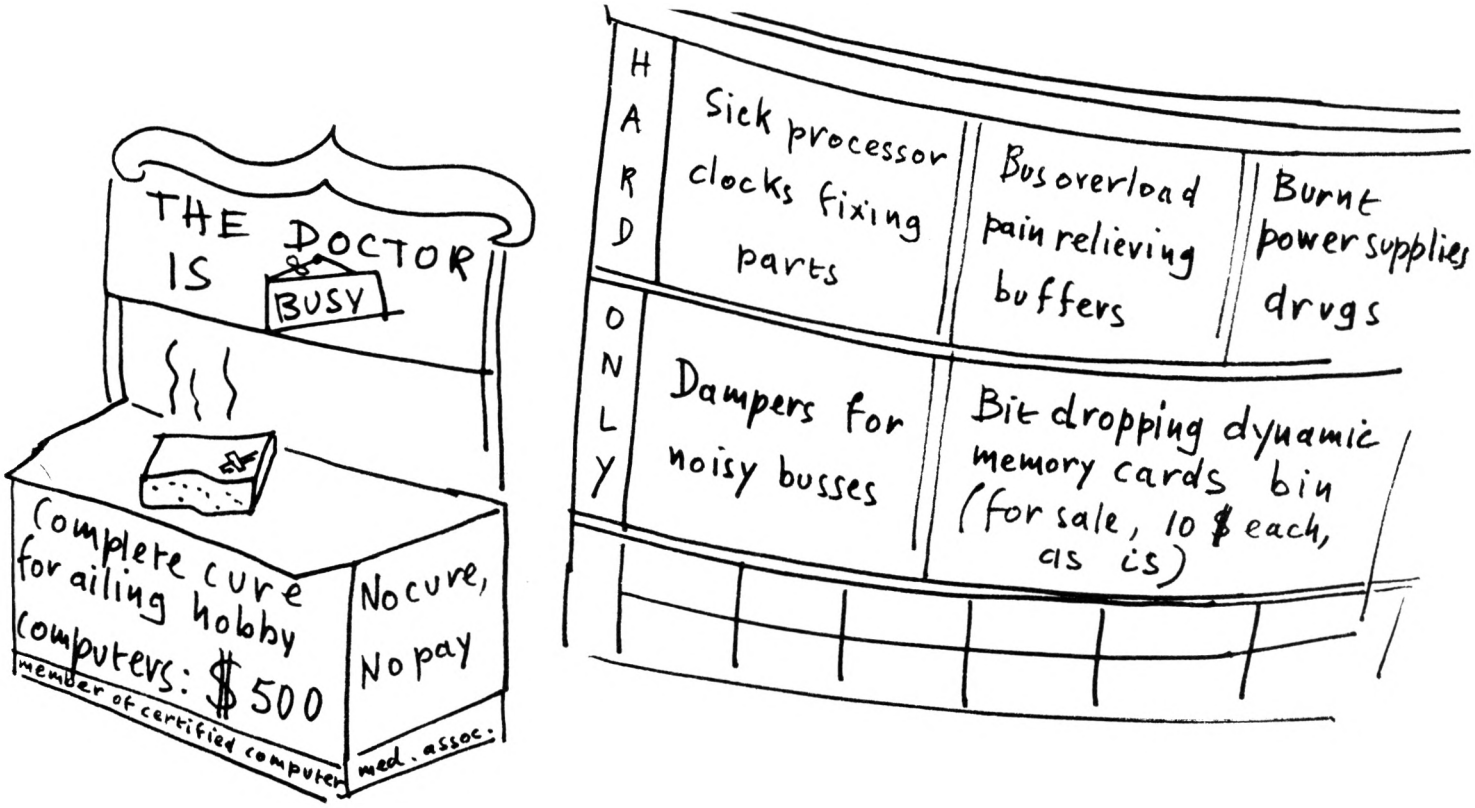
des signaux obtenus sur le bus est essentielle. Elle dépend non seulement du dessin du bus, mais du bruit électrique généré par les diverses cartes de la machine, des charges capacitives, des conflits possibles de signaux, des bruits dus aux différentes alimentations électriques, etc.

Viennent ensuite les problèmes de circulation d'air et de refroidissement, de sensibilité aux parasites, aux variations de tension secteur, aux microcoupures, à l'électricité statique, à la température, etc., etc. Ce sont bien entendu toutes ces considérations, ajoutées à la qualité des composants de base qui constituent la machine, qui séparent l'ordinateur à utilisation professionnelle des machines d'amateur, de hobby, que l'on trouve sur le marché à bas prix.

8 - 2 - 1. Caveat emptor

Un mot sur ces dernières nous paraît s'imposer ici au passage. L'objectif de conception — très louable par ailleurs — des machines d'amateur, est d'arriver à mettre des ordinateurs à la portée des particuliers, et tout d'abord à leur portée financière. Ceci n'est pas facile et entraîne dès le départ, entre autres, une recherche d'éléments constitutifs au prix le plus bas, que ce soient circuits imprimés, connecteurs, alimentations ou même parfois les boîtiers intégrés eux-mêmes, (de nombreux fabricants écoulent par certains circuits de distribution des composants qui, bien qu'apparemment fonctionnant, ont été rebutés aux tests sur certains critères, tels que souvent une trop grande sensibilité aux variations de température ou de tension).

Notons un autre exemple précis : le problème des supports. Les supports de circuits intégrés donnent la possibilité de souder à la vague les circuits imprimés, ce qui est très économique. En effet, cette méthode de soudure, qui conduit à un très fort échauffement des pattes, ne peut guère être pratiquée directement avec les boîtiers sur



les circuits imprimés, on y met seulement les supports. L'insertion des boîtiers se fait après, une fois l'ensemble refroidi. Le matériel est alors testé, et les boîtiers de trop mauvaise qualité, défaillants dès le départ, pourront être changés en quelques secondes au lieu de devoir être laborieusement dessoudés à la pompe patte après patte. Mais l'ennui de tout ceci est qu'il y a support et support : les prix varient de 1 à 5 selon qu'on va du support estampé courant au support décolleté et doré, qui coûte — lui — dix fois le prix du boîtier TTL qu'il supporte ! Les premiers donneront à coup sûr de mauvais contacts dus à l'oxydation au bout de quelques mois de fonctionnement, compromettant gravement la fiabilité de la machine qui donnera des résultats erratiques, mais de façon non continue. Ces pannes intermittentes, ou même transitoires, sont des plaies qui empêchent bien évidemment toute utilisation professionnelle d'un matériel.

Il y a bien sûr une solution : l'emploi de boîtiers premier choix, et leur soudure directe sans supports, à la main, patte par patte (faible échauffement) au banc antistatique. C'est ce qui est fait pour Alcyane. Ouvrez d'ailleurs un autre ordinateur professionnel de qualité, un Hewlett-Packard par exemple : il n'y a pas de supports non plus.

Par ailleurs, le marché de naissance (USA) des machines de hobby est d'un dynamisme... qui va parfois un peu loin. Les ordinateurs passent de la planche à dessin à la vente en série en trois mois, avec des résultats prévisibles : un fonctionnement peu fiable. On voit alors d'étonnantes sociétés aux étonnantes petites annonces fleurir, telles que «si votre XXXX (ici une marque ou une autre d'ordinateur d'amateur bien connue) refuse de démarrer à froid le matin, ou s'il perd des bits dans sa mémoire, achetez nos kits de transformation n° yy et zz».

La plus connue de ces sociétés s'appelle... *Parasitic Engineering !*

Une concurrence effrénée aidant, l'industrie du plug-compatible, qui a mis si longtemps à apparaître sur nos gros ordinateurs, s'est développée ici en quelques mois. Le bus d'un des premiers ordinateurs de hobby à être vendu en réellement grande quantité, bus au dessin fort criticable d'ailleurs, est devenu la cible de cartes "compatibles" les plus variées luttant de prix, mémoires, interfaces, processeurs mêmes, provenant d'une cinquantaine de fabricants différents. Les amateurs constituent ainsi de véritables machines-cocktails : fort heureusement, la plupart d'entre eux sont munis d'un bon arsenal d'oscilloscopes double trace, de fréquencemètres ou d'analyseurs logiques, ce qui leur permet d'arriver à faire marcher ces assemblages hétéroclites qui doivent être "ajustés" un à un, car, si le bus est là, aucune

norme électrique ni de chronogrammes de signaux n'existe !

Régler les timings, ajouter des bufferisations, chasser les bruits — toutes excellentes distractions d'ailleurs que nous connaissons bien ! — permet d'arriver patiemment à une machine unique qui marche à peu près, à la grande joie de son propriétaire, qui aura par ailleurs beaucoup appris dans ce processus. D'où d'abondants échanges de courrier ou discussions dans des clubs très actifs (comptant parfois 8000 membres comme certain club provincial de la Côte Pacifique !) sur les marques de cartes mémoire qui "marchent" mieux, les astuces à employer, etc., etc.

Reste ensuite, une fois ce parcours du combattant achevé, à créer le software qui permette de faire fonctionner tout l'ensemble, mais ceci est une autre histoire, comme dirait Kipling...

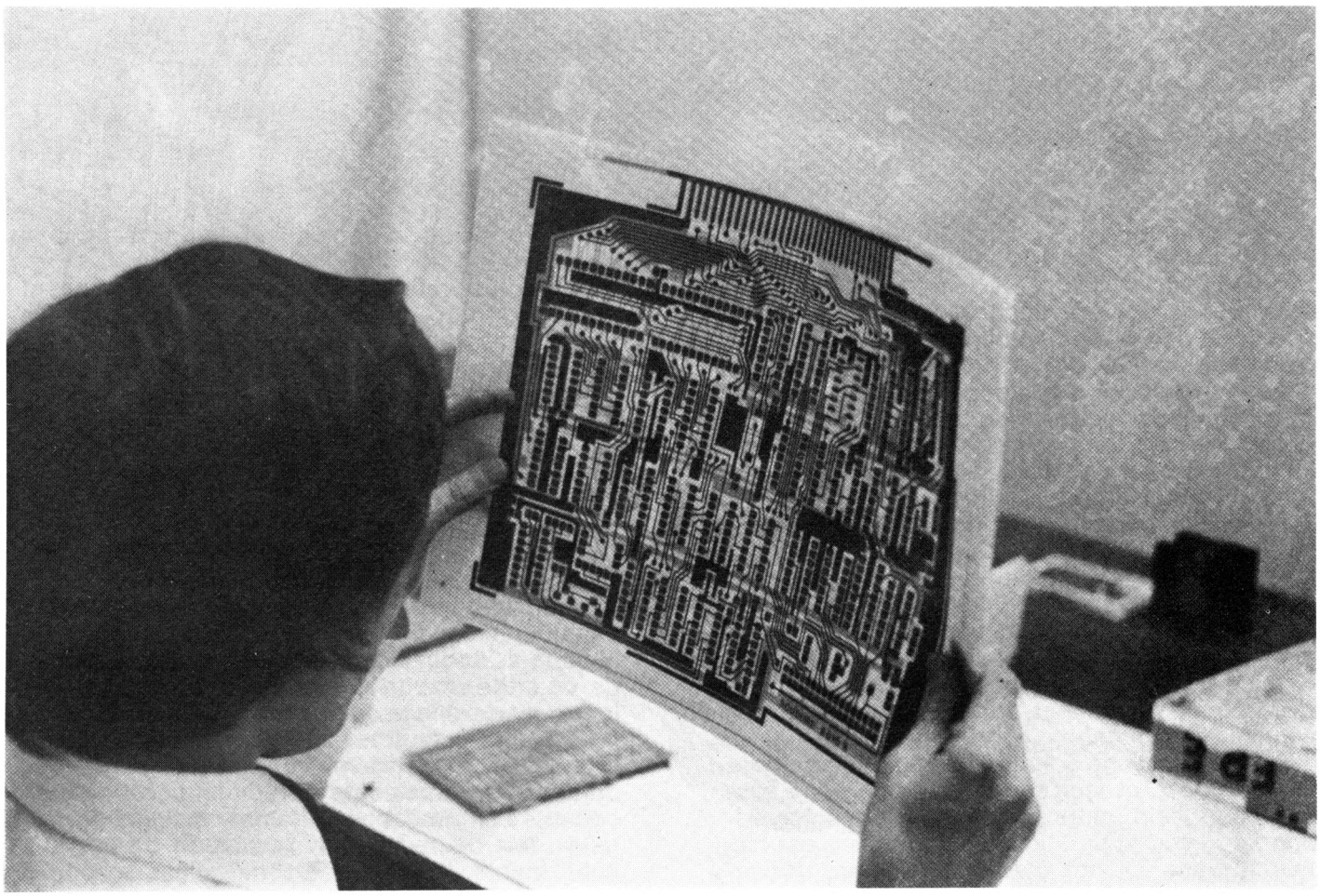
8 - 2 - 2. L'unité motrice

Mais revenons après ce détour à notre machine, professionnelle, elle.

Lorsque l'ensemble hardware/software a été amené à une homogénéité et à un niveau technique satisfaisants, on est alors en possession d'une unité de traitement universelle qui est le noyau de base, l'unité motrice. C'est elle que nous avons appelée Alcyane. C'est elle qui figure sur notre photo, vous l'avez reconnue.

Quelles sont les caractéristiques de cette machine ?

Sur le plan de la conception, l'une des principales options de départ est le choix de la taille des cartes. De grandes cartes, mettons de 30 cm sur 20, conviennent à une machine figée à application spécifique. En effet, on logera sur la carte



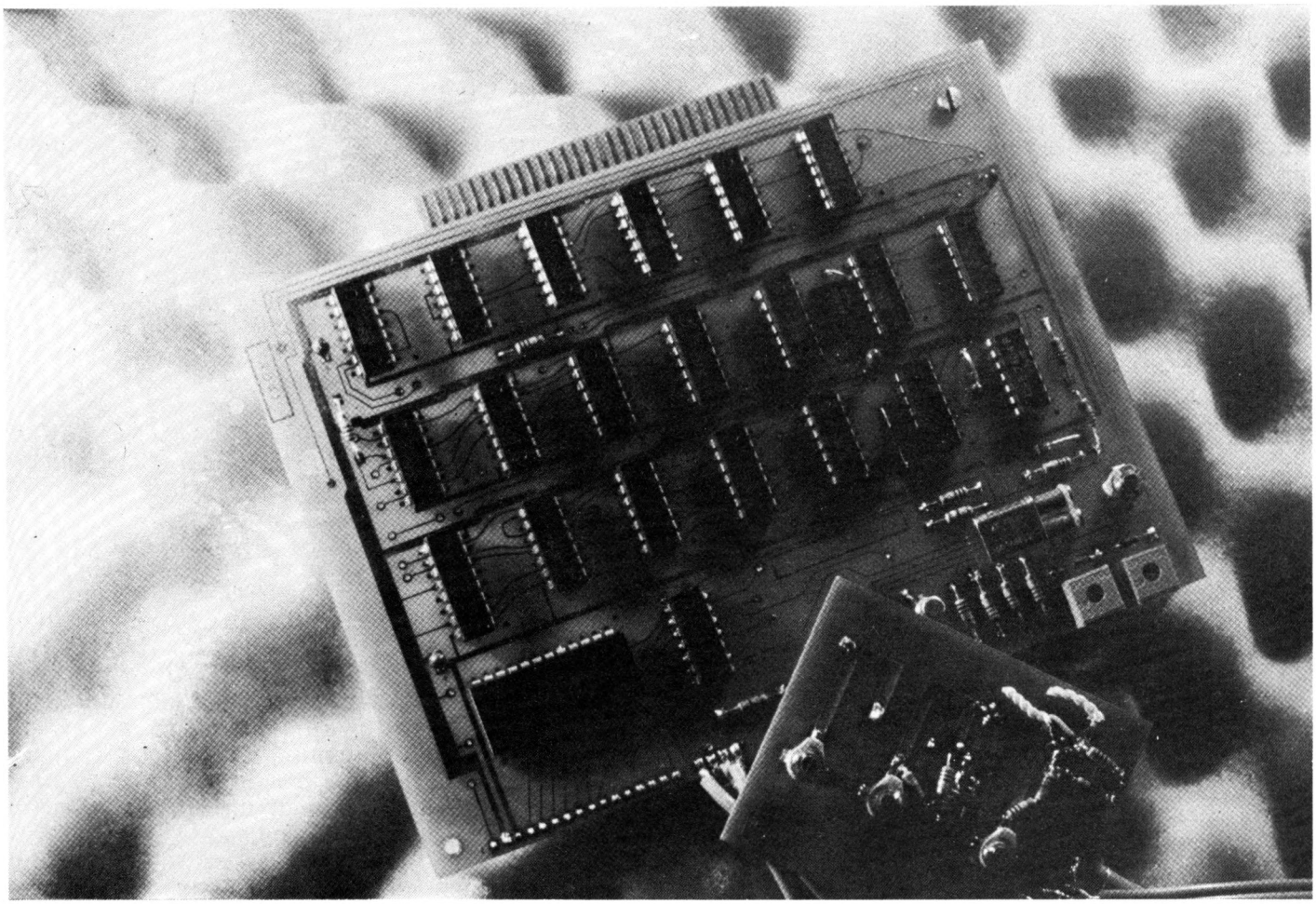
Inspection finale du masque d'un circuit imprimé (il s'agit d'un interface minidiskettes). Le «mylar» a été réalisé à l'échelle 2. Il sera réduit photographiquement à l'échelle 1 pour la fabrication. La qualité des circuits imprimés est fondamentale pour un ordinateur : conception, implantation, exécution graphique, qualité des matériaux et de la fabrication, tout importe.

les quantités de mémoire vive et de mémoire morte nécessaires pour l'application envisagée, ainsi que les interfaces spécifiques d'entrée et de sortie avec drivers de ligne, relais de puissance, etc., dont on aura besoin.

Au contraire, si l'on désire réaliser par exemple une machine de contrôle de processus industriel, dans laquelle on aura besoin d'un grand nombre de petits modules correspondant à des entrées/sorties très variables en nombre d'une machine à l'autre, on adoptera comme module de base de petites cartes, mettons par exemple de 8 cm sur 10 environ, qui figureront dans la machine à de nombreux exemplaires. Tout ceci influera naturellement beaucoup, aussi bien sur la conception du coffret et des supports physiques de cartes, que sur celle de l'alimentation, par exemple.

Une taille intermédiaire, permettant de loger une trentaine de boîtiers par carte, a été choisie pour Alcyane. Cette taille convient bien à une unité de traitement d'usage général, destinée aux applications de traitement de l'information courantes, aussi bien dans le domaine de la gestion que dans le domaine scientifique. On a un bon compromis entre l'économie de connectique apportée par les grandes cartes et la modularité nécessaire dans une machine. Chaque carte représentera donc un module que l'on enfiche sur le bus, module processeur, module mémoire vive, module mémoire morte, module interface d'entrée/sortie. Bien sûr, ce sont ces derniers qui sont les plus variés, vu la multiplicité d'organes périphériques que l'on peut désirer connecter à l'unité de traitement. Quelques commentaires s'imposeront donc à leur sujet.

Le microprocesseur nous ayant permis par l'économie de conception et l'économie de fabrication d'arriver à un coût très bas pour l'unité de traitement, le problème qui va se poser maintenant, va être celui d'essayer d'abaisser au maxi-



Un des modules de visualisation. Il comporte deux cartes (superposées, celle de dessous n'est visible que par son fichier à 62 contacts qui déborde, pour enfichage dans le bus). La troisième petite carte est un modulateur, qui ne sert que lorsque une connexion vidéo est impossible.

mum le coût des périphériques. Pour certains d'entre eux, il va être possible de faire quelque chose, et en particulier pour l'écran de visualisation.

Les matériels micro-ordinateurs étant, par nature, principalement destinés à l'informatique transactionnelle, avec traitement en temps réel, nous aurons toujours besoin d'écrans, et souvent à plusieurs exemplaires dans les systèmes temps réel un peu important. Une série d'interfaces a été développée, générant un signal vidéo qui permet l'affichage direct d'une zone de la mémoire centrale, sur moniteur vidéo ou récepteur de télévision.

Il va s'en suivre plusieurs avantages importants. Le premier, bien sûr, sera l'économie, puisque l'écran sera un appareil acheté sur le marché industriel courant, réalisé en grandes quantités et par conséquent d'un prix inférieur aux fabrications informatiques, qui, ne l'oublions toujours pas, ne sont pas à proprement parler des fabrications de séries industrielles. Le second avantage est que nous aurons, pour n'importe quel poste de travail, le choix de la taille de l'écran utilisé en fonction de la distance de vision, ou de tout autre critère. De plus, selon l'éclairement plus ou moins intense de l'environnement du poste de travail, nous pourrons choisir un écran à vision directe ou un écran à filtre fumé. Le troisième avantage est que la gestion de l'écran va être à la fois très facile et très rapide.

Facile, tout d'abord, puisque pour visualiser un message, il nous suffira par un simple "move" d'envoyer l'information de l'emplacement mémoire où nous venons de la générer, à l'emplacement mémoire correspondant à l'affichage sur l'écran.

Une telle opération est enfin extrêmement rapide, puisque le transfert de l'information se fait à une rapidité déterminée non pas par le débit d'une ligne de communication série, comme c'est le cas pour les écrans informatiques habituels, mais par le cycle de la mémoire centrale, de l'ordre de quelques centaines de nanosecondes par octet en général. Par exemple, il sera possible de substituer la totalité des caractères affichés sur l'écran, en un temps qui sera inférieur au temps de masquage du spot entre le bas d'une image et le haut de l'image suivante. Le changement d'écran est par conséquent instantané pour l'utilisateur. Cet avantage est important dans le cas d'un dialogue direct avec la machine, mais il l'est aussi dans de nombreux types de systèmes, comme par exemple des systèmes d'affichage de messages, où dans une gare ou un aéroport par exemple, une unité de traitement peut mettre à jour les messages sur de nombreux écrans simultanément.

Bien entendu, une fois le "move" effectué, l'interface maintient l'affichage de façon absolument autonome en hardware, et le programme de l'unité de traitement se déroule librement pour se consacrer à d'autres tâches.

On dispose ainsi des possibilités d'affichage suivantes :

— affichage peu dense à grandes lettres très nettes suivant les dernières recommandations des groupes d'études européens des milieux du travail : 512 caractères par écran (16 lignes de 32 caractères) ; cette densité est en pratique adéquate pour la plupart des travaux de gestion courants, vu en particulier la rapidité avec laquelle on peut substituer un message à un autre ;

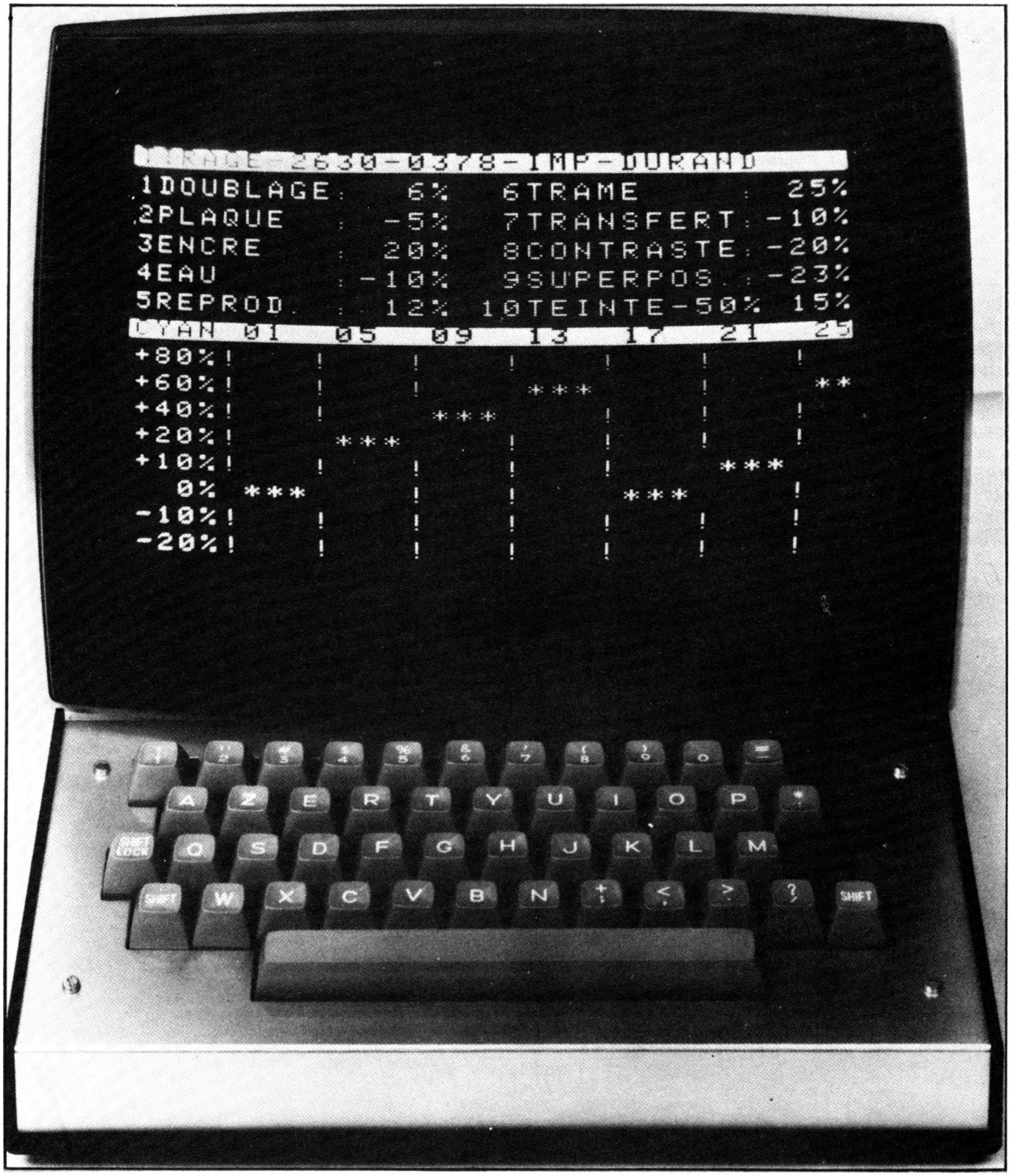
— affichage forte densité correspondant au standard informatique courant, 1920 caractères par écran (24 lignes de 80 caractères) ; cet affichage a été conçu spécialement pour le traitement de textes ; en effet il permet de visualiser non seulement les majuscules, mais les minuscules accentuées, avec les accents graves, aigus, trémas, cédilles, etc. ; là encore le fait de pouvoir choisir la taille de l'écran est important, on peut donc visualiser le texte prêt à imprimer, en y procédant à des retouches d'édition. Il va sans dire que ce travail est réalisé sur une unité de traitement qui, n'étant pas spécialisée, permettra de se livrer, une partie de la journée, à du traitement de texte, et le reste à des travaux classiques tels que facturation, comptabilité, etc. ;

— affichage graphique correspondant à une extension de l'affichage précédent : la résolution de cet affichage est de 65 536 points (256x256) ; on peut mettre plusieurs cartes graphiques dans l'unité de traitement ; avec une seule carte graphique on obtiendra un affichage avec deux niveaux de brillance, c'est-à-dire le noir et le blanc ; avec deux cartes on obtiendra quatre niveaux de brillance, et avec trois cartes, huit niveaux. L'affichage graphique est mixable sur l'écran avec l'affichage alphanumérique précédent, permettant par conséquent de superposer textes et dessins.

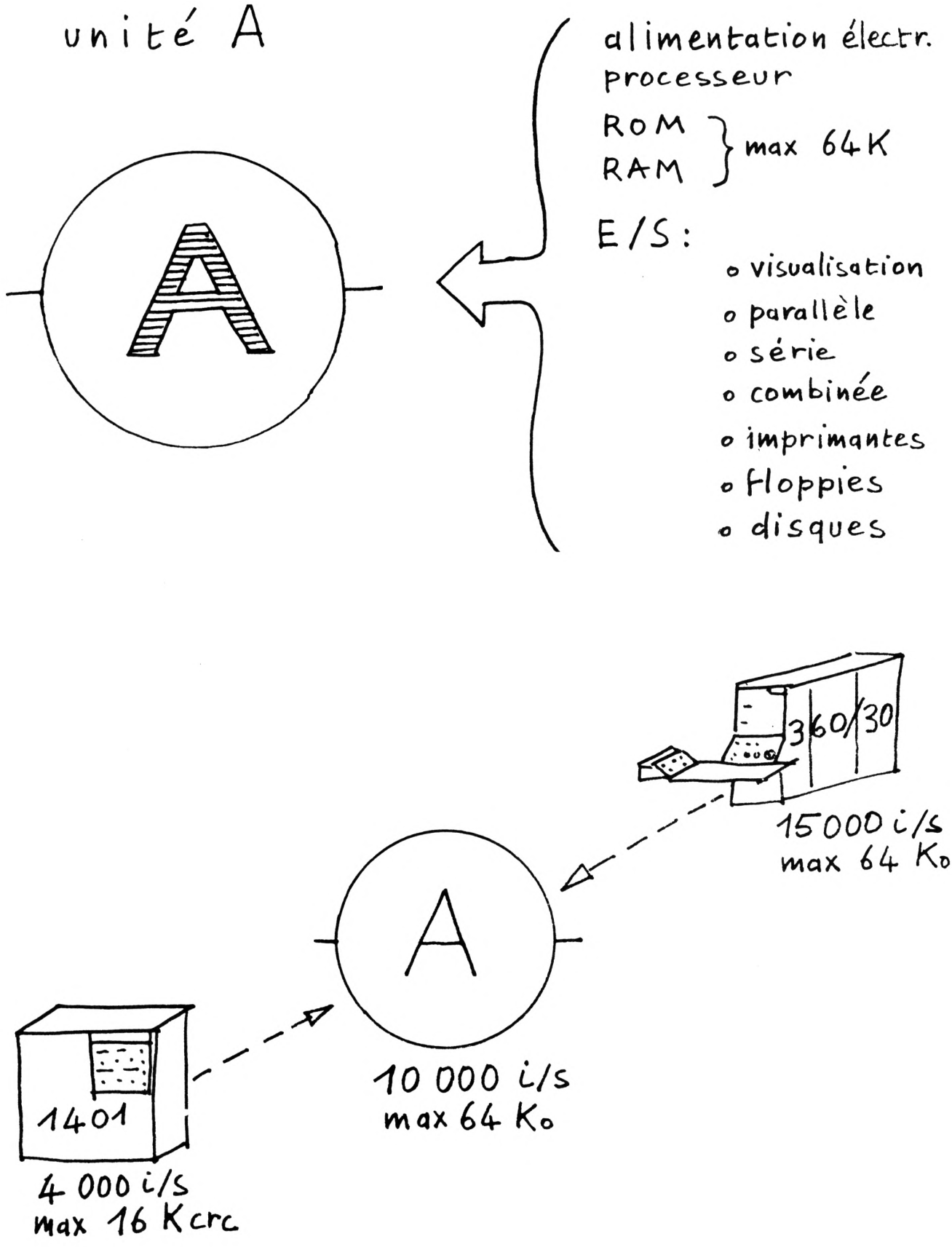
Pour conclure sur les modules d'affichage, ils ont été réalisés de façon à obtenir une définition élevée avec des caractères bien nets, aucune scintillation et une bonne possibilité de gestion en programmation.

Rappelons au passage que le fait d'utiliser des matériels d'affichage grand public, qui sont soumis à des essais très stricts, apporte une excellente sécurité sur la plan des radiations. Dernier point : il sera possible d'avoir en stand by des écrans de dépannage pour un prix additionnel très faible sur une installation.

Voici donc notre unité de traitement universelle Alcyane, que nous appellerons dans la suite unité "A". Une unité "A" c'est donc un ensemble autonome comprenant son alimentation électrique, avec les diverses tensions nécessaires au fonc-



Affichage donné par le module précédent. Il a été reproduit en pleine page pour faire apparaître la netteté et la stabilité de l'image. Cette photo directe de l'écran, non retouchée, a été posée plusieurs secondes, et la matrice 5 x 7 pointillée des caractères est cependant restée nettement visible. Le programme est utilisé par une imprimerie spécialisée dans les travaux en quadrichromie.


unité A
A
alimentation électr.
processeur
ROM
RAM
max 64K
E/S :
o visualisation
o parallèle
o série
o combinée
o imprimantes
o floppies
o disques
360/30
15000 i/s
max 64 Ko
A
10 000 i/s
max 64 Ko
1401
4 000 i/s
max 16 Kcrc

tionnement, une carte processeur, des cartes "ROM" (mémoire morte) et des cartes "RAM" (mémoire vive) jusqu'à concurrence de 64 Ko, ainsi que des cartes d'entrée/sortie choisies parmi les suivantes (la liste n'est pas complète, nous avons omis les cartes trop spéciales) :

— sortie visualisation (2 types, 6 modèles) ;
— entrée/sortie parallèle (3 types : universelle, à 24 lignes d'entrée/sortie programmables pour l'utilisation en assembleur, entrée/sortie 8 bits pour utilisation en Basic, entrée/sortie rapide pour communication directe d'unité de traitement, dont nous verrons le rôle important par la suite) ;
— entrée/sortie série (3 modèles : CCITT V 24 tension pour l'utilisation de la plupart des périphériques courants disponibles avec interface série de ce type, ou interface RS 232 qui est très voisin, entrée/sortie série modem, pour les transmissions de données en mode asynchrone, entrée/sortie boucle de courant pour la connection de périphériques plus anciens, tels en particulier que certains télé-imprimeurs) ;
— interface combiné permettant l'entrée des informations par le clavier, l'entrée ou la sortie série en local sur cassette, ainsi que la connection d'une imprimante de contrôle, sur papier métallisé ;
— sortie imprimantes (3 types, 7 modèles d'imprimantes, à aiguille, à marguerite ou à peigne, avec des vitesses allant de 45 caractères-seconde à 300 lignes par minute) ;
— enfin, entrée et sortie disques (4 types de disques, allant de 100 000 octets à 10 ou 20 méga-octets par unité, les supports étant soit des disques souples, soit des disques durs selon les capacités).

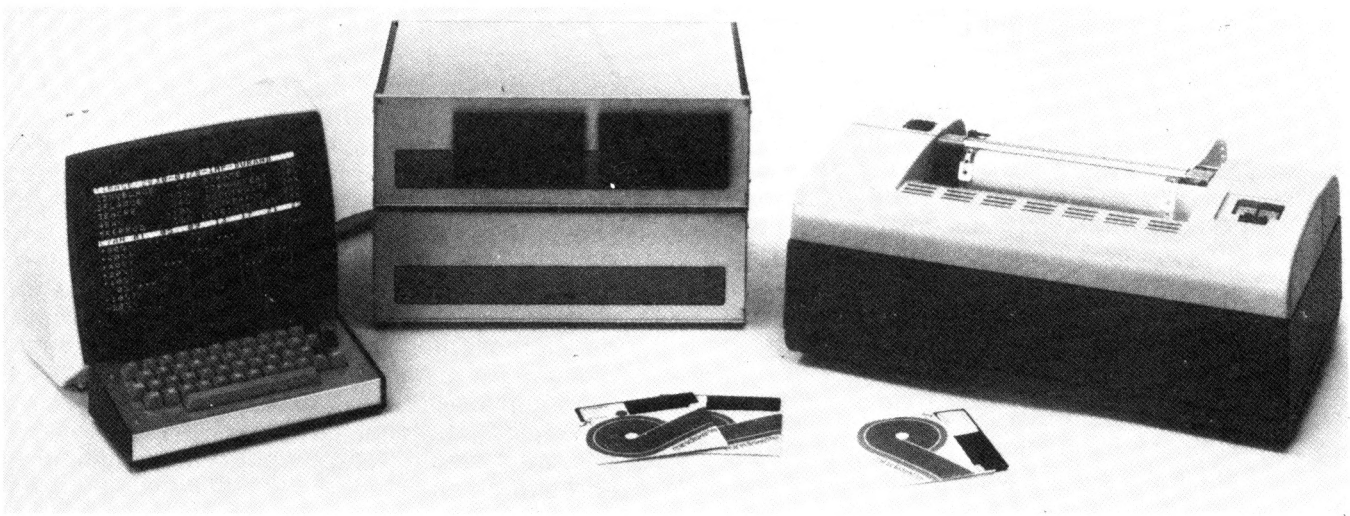
Le software de base? que ce soit l'Assembleur ou le Basic, est muni des drivers permettant d'actionner toutes ces entrées et sorties par des ordres de programmation simples.

Dès lors, l'unité "A" est devenue un système.

Question maintenant : où se place "A" dans le spectre des ordinateurs tels que nous les connaissons sur le marché !

Pour cette comparaison nous allons prendre une unité "A" standard, avec l'horloge normale à deux mégahertz. Dans ce cas le processeur exécute environ 200 000 instructions par seconde, avec des cartes mémoire à 600 nanosecondes de temps d'accès par octet. Cette vitesse ne doit pas être comparée à un nombre d'instructions par seconde mesuré sur gros ordinateur, les instructions de ces derniers étant plus complexes, et accomplissant davantage de travail qu'une instruction de microprocesseur : en particulier toutes les opérations décimales sur chaînes de caractères en mémoire qui sont monnaie courante, aussi bien en application de gestion qu'en application scientifique, sont systématiquement réalisées par des sous-programmes lorsque l'on utilise des microprocesseurs. Nous rappellerons simplement les résultats globaux obtenus dans les chapitres précédents : un gibson mix de gestion appliqué à l'unité "A" donne une vitesse moyenne de 10 000 instructions par seconde. Ceci place donc l'unité "A" entre le 1401 et le 360/30, qui ont été les chevaux de bataille du plus grand constructeur sur le marché général en seconde et en troisième génération.

Un coffret "A" mesure 45 cm, pèse 8 kg et est vendu environ 16 000 FF avec 22 K de mémoire, clavier et écran (prix unitaire 1978). Il sera la base de notre *architecture nodale*. Nous remarquerons



L'installation de base. C'est une configuration standard d'usage étendu, pour le travail en conversationnel à l'écran, les fichiers sont en accès direct sur minidiskettes. Mille lignes de facturation/stock par jour passent sans problème. Sept types d'imprimantes sont connectés.

donc au passage que le prix d'une unité de traitement est maintenant sensiblement divisé par cent, par rapport à ce que nous avons pu connaître il y a peu d'années. Des formes entièrement nouvelles d'architecture de machine sont donc d'ores et déjà possibles.

8 - 2 - 3. Cavalier seul

Naturellement, la toute première utilisation est d'employer une unité "A", toute seule. Nous voyons ici un système de base avec clavier/écran, deux mini-disques souples et imprimantes à aiguilles. Ce système vaut 35 000 F. C'est la configuration standard de la petite entreprise, du service ou de l'agence de grosse entreprise, du laboratoire, du cabinet de profession libérale ou de l'établissement d'enseignement. Ses possibilités sont considérables, elles sont surtout limitées par les périphériques plus que par l'unité centrale. Bien que nous ne pensions pas que la vocation d'un micro-ordinateur soit d'effecteur systématiquement des travaux de masse, nous ne pouvons néanmoins nous empêcher de dire que de nombreuses entreprises réalisent par exemple des travaux de facturation/tenue de stock avec traitement complet d'un millier de lignes par jour sur des systèmes de ce type.

Mais augmentons la capacité disques : pour 15 000 F de plus, nous aurons plus d'un million d'octets en ligne sur la machine. Les possibilités deviennent considérables, et bien entendu on diminue beaucoup les manipulations de diskettes.

Rappelons au passage, pour ceux des programmeurs qui se demandent ce que l'on peut faire avec des unités dont les tailles de mémoire en Ko s'échelonnent de 20 à 60, que les 15 000 et quelques 1401 qui ont été installés dans le monde, avaient une mémoire de 4 à 8 K en général, et 16 K pour un petit pourcentage du parc installé. Ici, avec l'unité "A" la segmentation des programmes est très facile, et un seul ordre suffit en Basic pour appeler des morceaux successifs de programmes à exécuter à partir des diskettes si le besoin s'en fait sentir.

Enfin, pour 10 000 F de plus, on fait le bond à un système ayant 10 millions d'octets en ligne sur une cartouche de disque dur interchangeable, utilisant la nouvelle technologie à servo-traquage, mise au point par les ingénieurs de l'usine Bull de Belfort. Nous reviendrons plus loin sur la conception du contrôleur qui a été réalisé pour cette unité, conception qui sera à nouveau une illustration des possibilités de la standardisation d'une unité de traitement "A".

8 - 2 - 4. Bis repetita placent

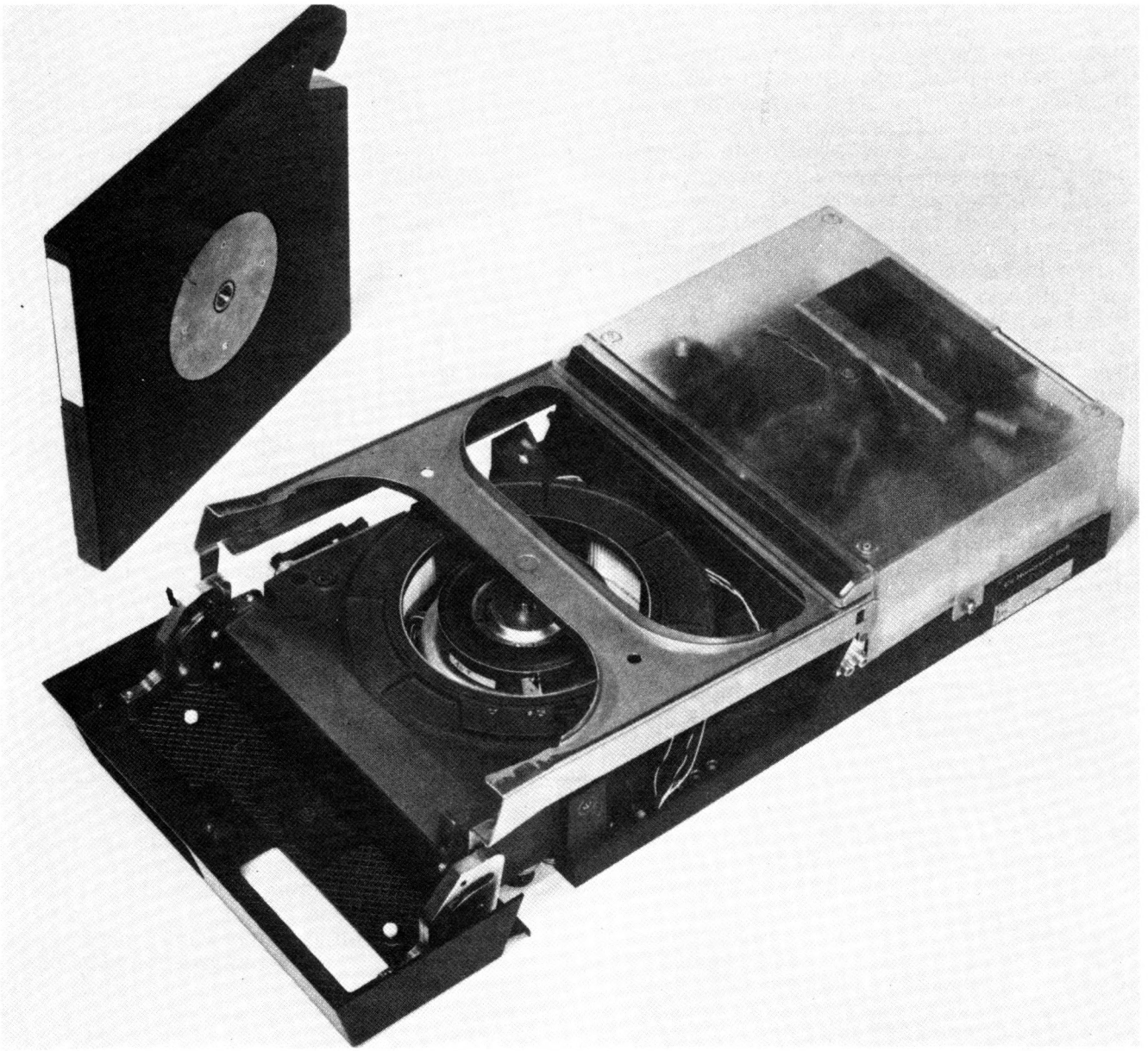
Quittons maintenant le domaine de l'unité "A" seule, pour faire notre première escalade, deux claviers et deux écrans.

Cette configuration correspond, on le constate expérimentalement, à une forte demande du marché. En particulier, dans de nombreuses PME, la machine étant installée au bureau administratif, le responsable de la PME désirera cependant avoir sur son bureau un clavier et un écran, pour pouvoir interroger à tout moment un fichier stock ou l'état d'un compte client, par exemple. Il est possible de réaliser cette configuration pour 3 000 F de plus.

Principe utilisé : le module racine d'un programme Basic, va lire au moyen de l'instruction «KEY IN» les chaînes de caractères qui sont en provenance des claviers. Si le programme correspondant au traitement de ces chaînes n'est pas en mémoire, il sera appelé en overlay à partir des disques. On réalise donc une multiprogrammation extrêmement simplifiée (qui en fait n'en est pas une, puisque de toutes façons l'unité "A" n'exécutera qu'un seul programme à la fois), mais ceci suffit amplement pour les besoins de la cause. Le module racine Basic dispatchera les réponses vers les écrans par une simple instruction «WRITE» utilisant le «FORMAT». On dispose donc ainsi d'un système conversationnel à deux postes convenant bien à un petit trafic transactionnel, et ceci avec une incontestable économie de moyens.

Mais voyons un peu maintenant les configurations qui sont réalisables autour de systèmes disques, principalement les mégadiskettes ou les disques rigides rapides. Le contrôleur des disques rigides... c'est encore une unité "A", qui va être transformée en unité spécialisée, assurant la gestion des disques. Cette unité "A" va simplement être munie d'un module interface spécial, travaillant en accès direct mémoire, et qui va permettre de faire transiter, en provenance ou à destination des disques, l'information qui circule à la cadence rapide de 8 mégabits par seconde. Nous munirons en outre l'unité "A" de mémoire ROM (4 K), et de mémoire RAM (4 K également). En ROM nous aurons les routines de base de lecture et d'écriture sur disque, avec accès direct par nom symbolique de fichier, et adressage relatif (accès au nième enregistrement du fichier). Ce mode d'accès est le même que pour les diskettes, et les programmes écrits pour ces dernières passeront sans modifications. La mémoire «RAM» servira, elle, de mémoire tampon pour faire transiter des enregistrements.

Notre unité "A" est donc devenue une unité de traitement spécialisée. Sur le croquis correspondant, elle est repérée par l'indice 2 : c'est l'unité

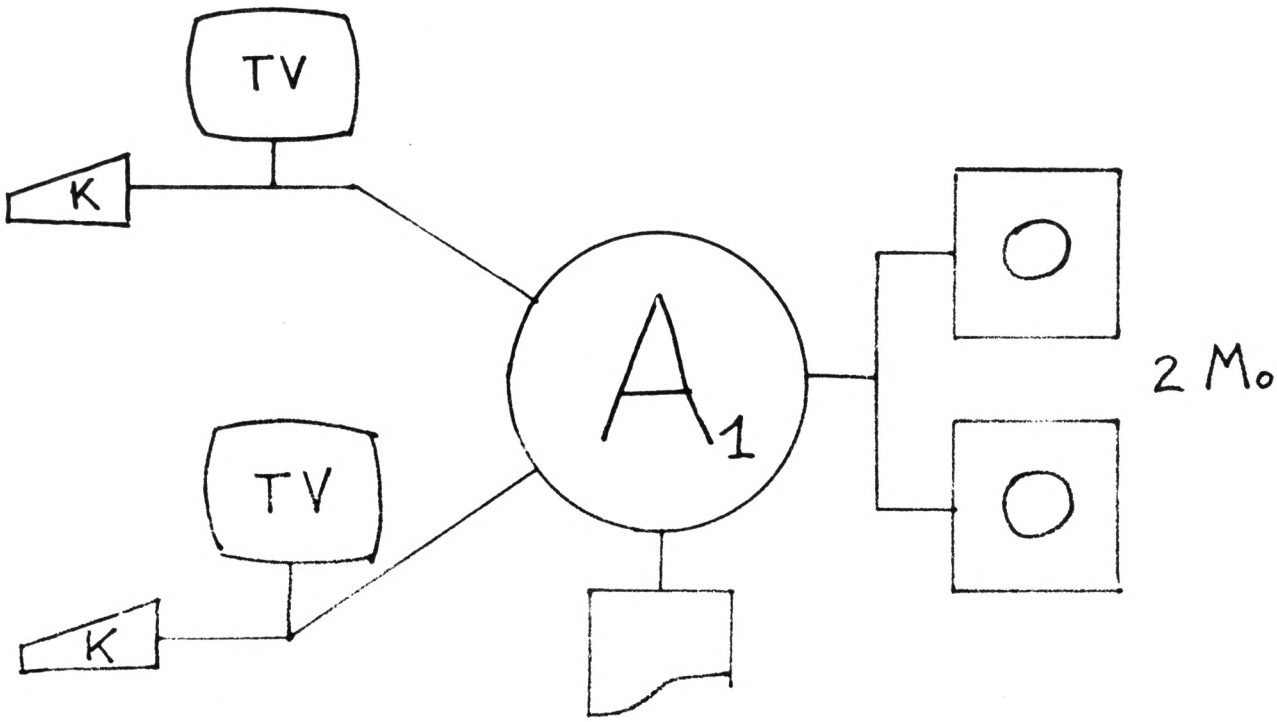


La nouvelle unité Cynthia à servo-traquage des ingénieurs de Bull. On voit le dessous de la cartouche amovible hermétique de 10 méga-octets. Le moteur d'entraînement, pleinement visible, a un faux rond inférieur à deux microns. Le positionneur à bobine mobile (voice coil) se devine à travers le boîtier de plastique translucide. La mécanique est réduite à presque rien par l'utilisation d'une électronique élaborée.

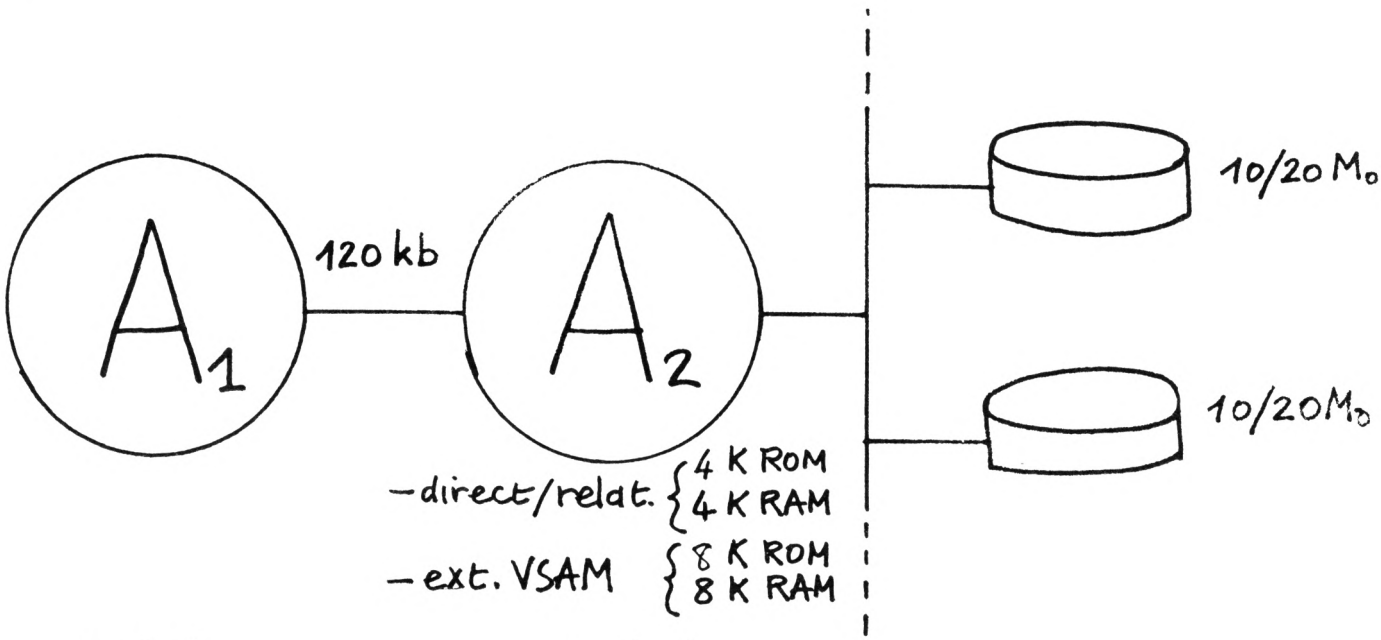
A2. Nous avons donc ainsi obtenu un puissant contrôleur intelligent qui peut gérer jusqu'à quatre disques durs. Le transfert entre l'unité "A" de traitement (A1 sur le croquis) et l'unité A2 s'effectuera sous forme parallèle par les cartes interfaces rapides de communication unité de traitement à unité de traitement, qui seront gérées par les instructions Basic du côté A1 et par le programme en ROM du côté A2. Si notre unité A2 est devenue une unité de traitement spécialisée, elle utilise toujours les mêmes modules de base. La vitesse de transfert entre les unités est de 120 000 bauds environ. Si nous supposons des enregistrements de fichiers de 250 octets par exemple, le transfert de l'enregistrement de A2 vers A1 ne demandera que deux centièmes de seconde, et la demande de l'enregistrement de A1 à A2 un temps considérablement plus court.

On voit que la charge de transmission des unités est très petite vis-à-vis des temps de travail au clavier ou des temps de recherche sur disque.

Si maintenant nous étendons les tailles de ROM et de RAM de A2, une nouvelle possibilité très intéressante apparaît : celle d'y loger complètement



BASIC → KEY IN dispatcher



unité de traitement

unité de traitement spécialisée

la méthode d'accès disque. Cette fois, nous pourrons gérer nos fichiers entièrement à partir du contrôleur en employant une hiérarchie de fichiers index, conçus selon les bonnes méthodes de Knuth, voisines du VSAM classique.

Le principe de base sera le suivant (sans rentrer dans les détails) : un fichier index parallèle au fichier principal est créé automatiquement. Chacun des enregistrements du fichier index comprend l'indicatif de base de l'enregistrement du fichier, qui sera par exemple le code de pièces détachées, s'il s'agit d'un fichier pièces, accompagné d'un pointeur indiquant l'adresse physique de l'enregistrement pièces sur le disque. Ce fichier index est découpé en blocs. Un second niveau de fichier index comprendra un enregistrement pointeur pour chacun des blocs du niveau hiérarchique inférieur, et ainsi de suite jusqu'à ce que l'on arrive à un niveau hiérarchique suffisant pour que l'ensemble des pointeurs tienne dans un seul bloc physique, lequel sera logé dans la mémoire RAM de A2.

Lors d'une demande d'accès émanant de A1, une recherche sera d'abord faite dans le bloc en RAM de A2 pour y trouver un pointeur, lequel permettra d'amener à partir du disque le bloc du niveau hiérarchique suivant du fichier index pour y rechercher à nouveau un pointeur, et ainsi de suite jusqu'à ce que l'on arrive à l'adresse physique de l'enregistrement sur disque.

Le temps de recherche totalement aléatoire sur l'unité de disques D 120 Cynthia de Bull est de l'ordre de 70 millisecondes, mais bien sûr il est en pratique inférieur vu l'organisation physique des fichiers index sur le disque. Le nombre d'accès nécessaire dépend de l'importance du fichier et des tailles de blocs physiques. Il est usuellement de l'ordre de trois ou quatre accès, ce qui nous amènera donc à deux dixièmes de seconde pour atteindre un enregistrement.

On peut accéder à un enregistrement non seulement par l'indicatif ou clé primaire, mais également par des indicatifs secondaires. Ces derniers sont suffisamment nombreux pour que dans la pratique tout fichier puisse être manipulé sans avoir recours au tri. C'est une simplification considérable qui est rendue possible en pratique du fait que la rapidité d'accès des disques, jointe à la lenteur relative des périphériques d'impression, fait qu'il n'est pas rentable de trier des informations, et qu'il vaut beaucoup mieux y accéder directement à chaque fois en fonction des critères de recherche. Une seule restriction bien sûr : il n'est possible de créer de nouveaux enregistrements dans le fichier qu'à l'aide de l'indicatif de recherche principal.

Ainsi donc, notre conception d'unité de traitement universelle nous a permis de réaliser facilement un *processeur de fichier* complet. Que pouvons-nous en faire ?

8 - 2 - 5. L'architecture nodale

Les principes de l'informatique répartie peuvent donc être appliqués non seulement à l'intérieur d'une entreprise, mais à l'intérieur même de la structure d'une machine. C'est ce que nous nous sommes efforcés de faire.

Le croquis suivant représente une configuration, en fait la première configuration de multiposte d'Alcyane qui ait été livrée en clientèle. Cette configuration convient bien à un trafic transactionnel moyen, pour lequel on recherche un système particulièrement économique. Dans ce système en tandem, l'unité A0 comporte en ROM un programme de scrutation de claviers et de transmission qui a le modeste encombrement de 3 K, et en RAM un tampon de 1 K. Si l'on ajoute un quatrième clavier, le tampon passe à 2 K. Nous sommes donc ici en présence d'une autre forme d'unité de traitement spécialisé, *processeur de claviers/écrans.*

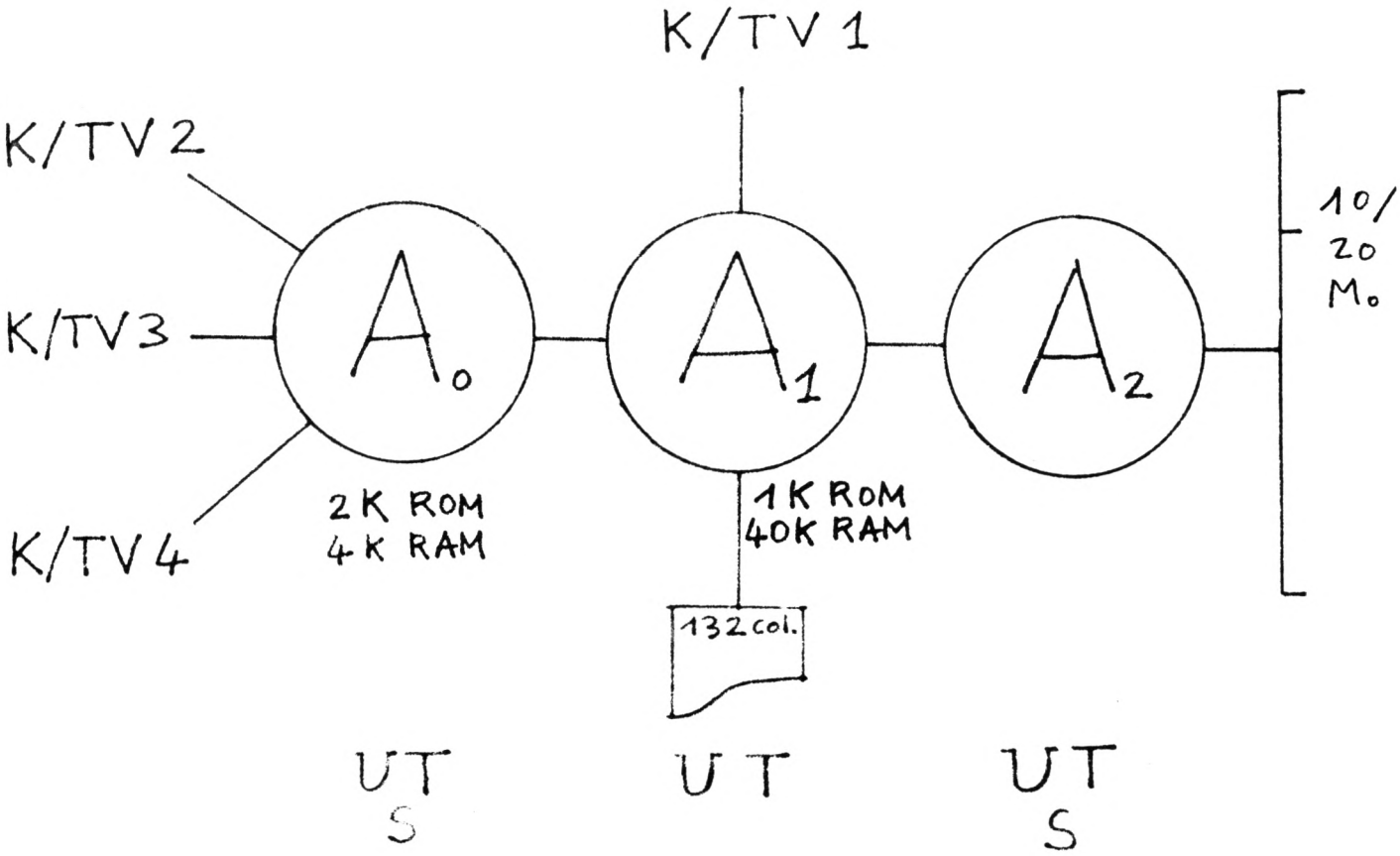
On voit clairement que, profitant du faible coût d'une unité de traitement "A", on utilise son énorme excédent de puissance pour remplacer des fonctions câblées par un programme en ROM, puisqu'il est devenu économiquement possible de se permettre le luxe de consacrer un 360/30 entier à gérer quelques claviers !

Résultat : fini l'operating system. En effet, dans de tels systèmes multiprocesseurs, les unités dialoguent entre elles au moyen de petits programmes spécialisés simples, chaque unité travaille en monoprogrammation, mais tout le monde en même temps. Il n'y a donc plus de nécessité de réaliser un système d'exploitation, et c'est le fait de pouvoir se débarrasser de lui qui donne la liberté de *l'architecture nodale.*

A titre d'indication, le prix de ce système, dont la puissance globale est déjà importante avec quatre claviers/écran, 3 + 1 K en A0, 40 K en A1 et dix mégaoctets disque dur gérés par A2, plus une imprimante 132 colonnes, est de 115 000 F.

Pour un trafic transactionnel plus intense, on adoptera une solution radicale qui est présentée par le croquis suivant : tout simplement une unité "A" normale tout entière pour chaque poste, exécutant directement le programme de traitement demandé, segmenté ou non à partir des disques.

Cette fois-ci, au fur et à mesure qu'on va ajouter des postes dans l'installation, on va accroître simultanément la puissance totale de traitement du système, et la taille mémoire centrale du système. Par exemple, dans un ensemble à cinq postes, on aura une mémoire centrale répartie qui pourra aller en pratique, de 130 à 300 K. Le système à 130 K pourra comporter par exemple, 22 K par poste et autant dans le contrôleur, deux disques à cartouches interchangeables de 10 mégaoctets chacun, et deux imprimantes pour un prix total de l'ordre de 200 000 F. Si les programmes sont



complexes et que l'on désire éviter une charge de segmentation qui peut devenir importante, pour 80 000 F de plus, on passera à une mémoire centrale répartie de 300 K.

On voit l'effet économique considérable qui est amené par *l'architecture nodale* à base de microprocesseurs. Nous sommes en effet en présence, ici, de systèmes dont la puissance de traitement et la taille mémoire les placent en plein dans le domaine des gros minis ou des ordinateurs de taille moyenne, qui plus est, entièrement tournés vers le travail transactionnel temps réel. Le seul point qui n'est pas présentement au niveau des ordinateurs de taille moyenne est la capacité de stockage fichiers ; ce n'est qu'une question de temps, dans moins d'un an, des unités de disques de 120 millions d'octets, adaptées au micro-ordinateurs, vont être disponibles.

Comment peut s'établir dans un système de ce genre, le bilan de l'occupation du contrôleur, qui est manifestement le seul point qui peut provoquer un ralentissement de débit ? Typiquement, sur les cinq postes, on peut penser que trois seront utilisés pour des travaux soutenus, et les deux autres pour des interrogations ou des mises à jour de fichiers à des cadences moins grandes. Si les trois premiers postes travaillent, par exemple, à raison de 1 500 lignes de transactions par jour chacun, que se passe-t-il ? On a en général besoin d'accéder à un fichier par groupe de transactions, ce sera par exemple au fichier clients pour les diverses lignes d'une commande client ou d'une facture, et à un autre fichier au niveau de chaque transaction, par exemple le fichier produits.

En cinq heures de travail effectif par jour à un poste, on voit que le temps moyen de traitement d'une transaction sera par exemple d'une quinzaine de secondes, temps dans lequel la majeure partie sera consacrée à la frappe au clavier. Les accès fichier représenteront un temps de l'ordre de la demi-seconde en gros par transaction, soit une seconde et demie pour l'ensemble des trois postes à gros débit. On voit que dans ces conditions la charge du contrôleur "A" sera de l'ordre de 10 %, avec les paramètres cités.

Si nécessaire (transactions très courtes et nombreuses nécessitant de nombreux accès à divers fichiers), on peut augmenter encore le trafic en utilisant cette fois plusieurs contrôleurs, chacun gérant un certain nombre de fichiers. Sur le croquis correspondant, dans lequel nous avons figuré deux contrôleurs, la notion de *réseau nodal* apparaît pleinement. Les disques qui figurent

dans cette configuration sont des unités doubles comprenant deux disques de dix millions d'octets dans chaque unité. On voit que dix unités de traitement fonctionneront simultanément dans un tel système. Le prix total du système est inférieur à 400 000 F.

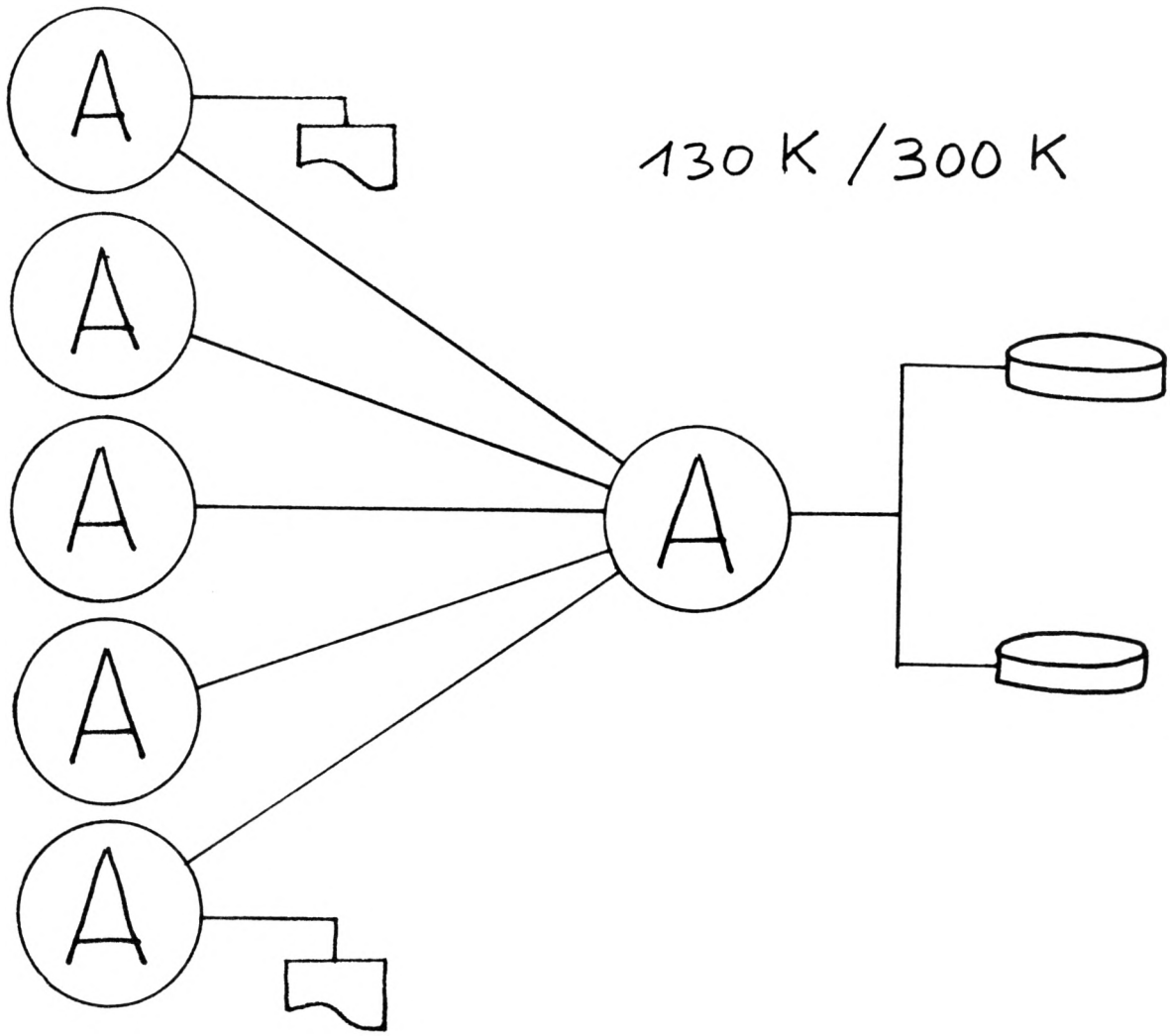
Mais cette configuration correspond à un cas extrême peu adapté au micro : gros programmes, transactions nombreuses et courtes, multiples petits fichiers. Dans les cas courants ce sera plutôt l'inverse. Comment pourra-t-on alors architecturer le système ?

La réponse est simple : nous disposons déjà des *sous-ensembles* de construction modulaires désirés, tout prêts. Revenons en effet à notre première configuration multipostes, avec le processeur de claviers/écran A0 et l'unité de traitement A1. Coupons entre cette dernière et le processeur de fichiers A2 : nous avons à gauche de la coupure un *sous-ensemble* à quatre ou cinq claviers qui est justement conçu pour discuter avec le processeur de fichiers. On peut donc l'insérer à volonté dans la configuration.

Sur ce croquis, on constate une autre évolution du système, le dédoublement du processeur de fichiers en deux unités : la première prend en charge la gestion des communications avec les diverses unités de traitement, la seconde gère les fichiers comme déjà vu.

Sur le plan sécurité, de nombreuses remarques sont à faire. Notons tout d'abord que chaque disque possède sa propre alimentation. Il en est de même de chaque unité "A". Elles sont toutes autonomes, possèdent leur propre alimentation électrique, leur propre processeur. Par conséquent, le fonctionnement en mode dégradé, qui est si difficile à planifier dans un ordinateur conventionnel, se présente ici de manière tout à fait évidente. On bénéficie par conséquent d'une sécurité de fonctionnement considérable.

Par ailleurs, pour 5 % seulement d'investissement supplémentaire, on pourra disposer en



stand-by par exemple d'un troisième contrôleur de disques, de deux écrans et d'un coffret d'unité "A" comprenant une alimentation. On peut alors s'interroger sur le meilleur compromis financier, entre le contrat de maintenance classique d'une part, et l'achat de matériel supplémentaire d'autre part, qui permet alors de se contenter simplement de renvoyer au constructeur, par service rapide de chemin de fer, le matériel défaillant pour dépannage dans le mois qui suit. L'économie engendrée par ce type de maintenance, maintenant devenu possible, est évidemment considérable. Enfin, envisagée au niveau d'un groupe régional de matériel, cette solution est encore plus intéressante.

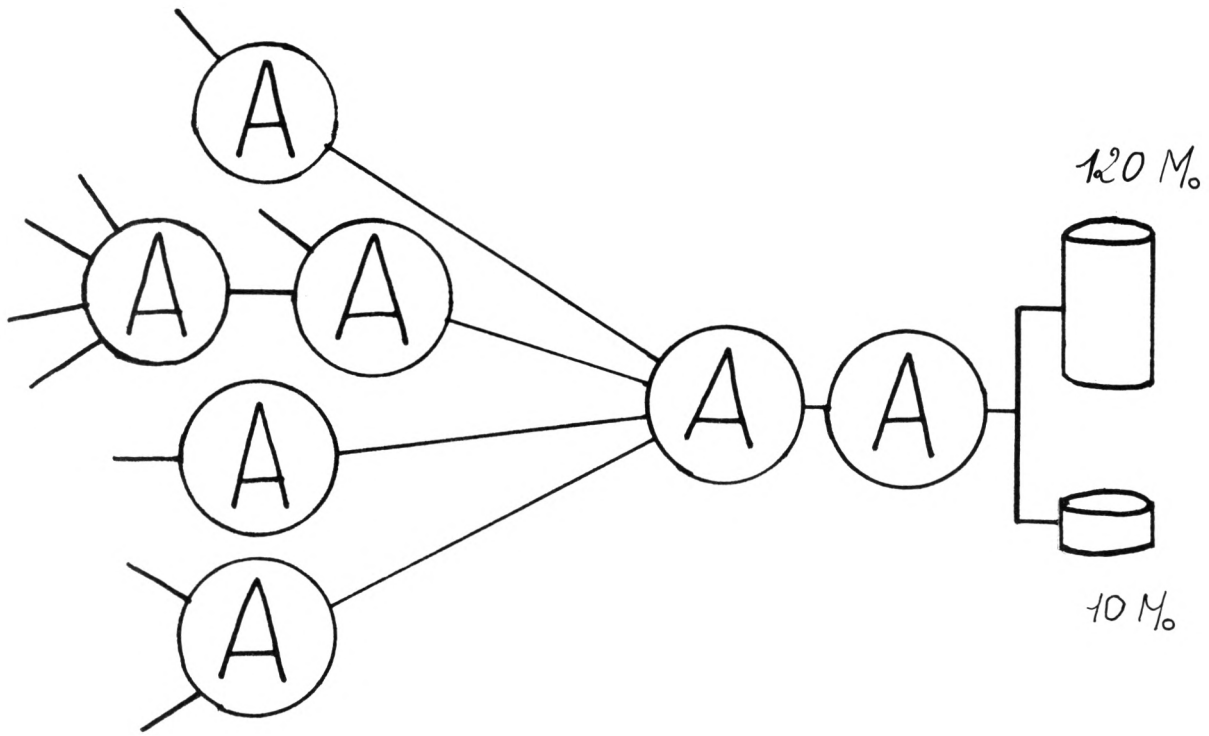
8 - 2 - 6. Le côté soft

Jusqu'ici, nous nous sommes surtout attachés à décrire le hardware de l'unité "A". L'aspect software est d'une égale importance, et la création de langages de programmation suffisamment souples et performants, s'étale nécessairement aussi sur un temps long. Trois niveaux successifs de langage évolué ont été conçus pour un spectre large d'applications allant de la gestion au scientifique. Il s'agit du Basic I, du Basic II et du Basic III.

Le premier est une version minimale de Basic, qui à son niveau déjà, comprend la gestion des chaînes de caractères. Le Basic II correspond à un niveau du langage que l'on appelle couramment Basic étendu, et comprend une trentaine d'instructions de plus que Basic I. Le Basic III, lui, comportant plus de 130 instructions et opérateurs, est un langage d'une souplesse maximale. Citons entre autres les caractéristiques suivantes :

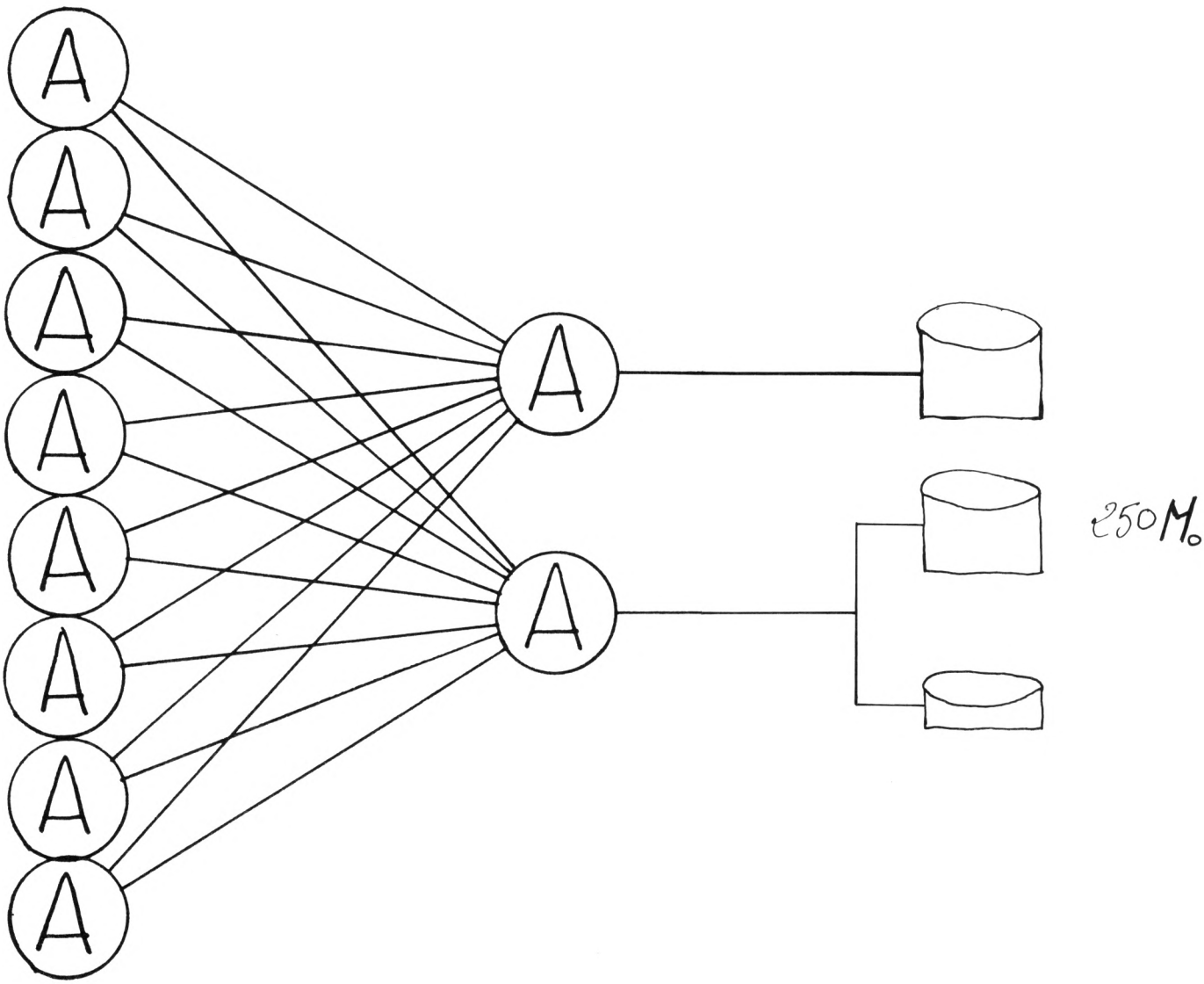
— orientation vers le travail à l'écran : pour ce faire, le Basic III comprend de nombreux ordres de gestion de l'écran, permettant de créer de façon commode des masques de saisie avec une grille à remplir, grille statique ou défilante, inversion vidéo de caractères, etc. N'oublions pas en effet que l'écran est le mode de contact privilégié avec l'ordinateur. Il est donc essentiel de pouvoir l'utiliser facilement pour créer l'échange conversationnel souhaité. Les formats d'écran pourront être stockés comme de simples enregistrements de fichiers, lus et envoyés sur l'écran par un ordre WRITE avec le format approprié par exemple. Des ordres tels que CLEAR permettent d'effacer sélectivement des zones de l'écran, TAB D permet de positionner très précisément un caractère sur l'écran en le superposant à un fond d'écran déjà existant, etc.,

— manipulation très complète de chaînes alphanumériques, soit des chaînes isolées, soit des chaînes en tableau. Nous touchons ici aux fondements mêmes des travaux de gestion, puisque les enregistrements de fichiers se présenteront sous la forme de chaînes alphanumériques, chaque rubrique de l'enregistrement étant une chaîne qui sera accolée à d'autres chaînes par concaténation, ou bien une sous-chaîne faisant partie d'une chaîne beaucoup plus longue, qui sera à ce moment-là l'enregis-



trement complet. On peut à tout instant extraire une sous-chaîne quelconque à l'intérieur d'une chaîne par exploration à l'aide d'un SEARCH, transférer ensuite cette sous-chaîne dans une variable appropriée qui permettra le traitement, puis la réintégrer à l'intérieur de la chaîne initiale. Des instructions spéciales permettent de réaliser en particulier les contrôles, tels le contrôle de numéricité qui est nécessaire dans de nombreuses opérations de saisie. Il est ainsi possible d'effectuer ce contrôle cent fois plus rapidement que s'il était programmé par une boucle de Basic (dans la pratique, il s'effectue si vite qu'il passe complètement inaperçu). Enfin, une série d'instructions de conversion dans les deux sens permettent de passer de chaînes numériques à des chaînes de caractères ASCII, ou en binaire, ou en hexadécimal de façon à pouvoir faire face à toutes les situations particulières qui peuvent se présenter en cours de saisie, de transfert d'informations ou de traitement.

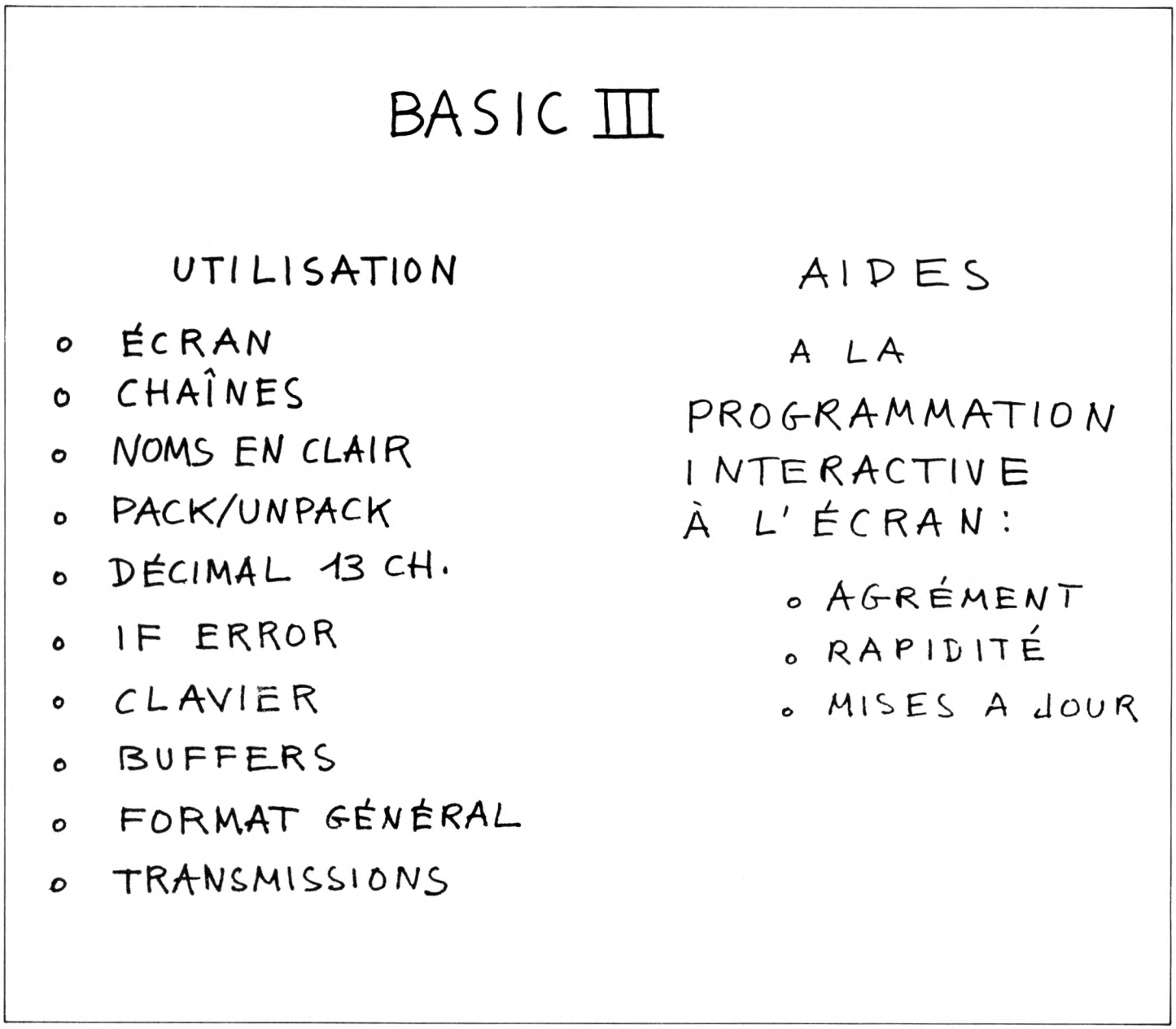
— Alors que dans les versions habituelles de Basic les noms de variables correspondent à une codification très stricte, usuellement sous la forme d'une lettre et d'un chiffre, ici les noms de variables peuvent figurer en clair, exactement comme pour du travail en Cobol. La lisibilité des programmes est ainsi considérablement améliorée, et si l'on se donne la peine de choisir des noms de variables suffisamment parlants et clairs, la majeure partie des commentaires nécessaires sur les listings peuvent être supprimés. Il devient en effet très facile de suivre le déroulement du traitement.
— Une possibilité très importante, c'est celle de pouvoir condenser les informations numériques sur fichier pour gagner de la place. Absente de



la plupart des versions de Basic, cette possibilité existe ici en Basic III, et peut bien entendu se conjuguer avec d'autres méthodes classiques de compression de fichier.

— La précision de calcul est également un point à considérer de près. En effet, pour les travaux comptables, il faut disposer d'une précision suffisante pour pouvoir effectuer des centaines d'opérations de cumul sans risque de déviation par des accumulations d'erreurs d'arrondi. A cet effet, la précision standard disponible est de treize chiffres significatifs, et les calculs sont directement effectués en décimal, supprimant les problèmes dus aux conversions. Il est possible par une commande spéciale de réduire cette précision dans les programmes qui ne la nécessitent pas, on gagne ainsi en place mémoire pour les variables, et en rapidité de calcul.

— Une instruction de récupération automatique d'erreurs à l'exécution, pour action correctrice, figure dans le langage. Ceci est important : si



une telle instruction n'existe pas dans le Basic, en exécution de programmes, dans le cas par exemple de certaines introductions erronées de données, le programme se débloquera avec un message d'erreurs à l'exécution, et il faudra prévoir une procédure manuelle de reprise qu'il sera nécessaire d'enseigner aux utilisateurs. Une instruction IF ERROR possédant en paramètre un numéro de ligne et un numéro d'erreur, permet au contraire par un sous-programme de traitement d'erreurs de déceler la nature de l'erreur qui vient de se produire en exécution, à la fois par le numéro de la ligne où elle a été détectée et le code qui permet d'identifier l'erreur. Une action correctrice peut ainsi être entreprise par le sous-programme de traitement d'erreur. Elle consistera le plus souvent en l'affichage d'un message destiné à l'opérateur.

— L'entrée au clavier étant une partie essentielle du travail pour l'utilisation de l'ordinateur, plusieurs instructions différentes ont été prévues pour pouvoir s'adapter aux différentes applications. Notons en particulier, outre l'instruction INPUT, qui est une instruction d'entrée classique en Basic, combinant à la fois l'affichage d'un message et de la chaîne de caractères qui vient d'être entrée, une instruction KEY IN permettant d'entrer à la volée des caractères un par un effectuant immédiatement une série de contrôles et de tests sur chaque caractère, sans attendre que toute la chaîne ait été frappée.

— Dans l'architecture nodale, il est nécessaire comme nous l'avons vu de pouvoir faire communiquer entre elles les unités de traitement. Ceci est réalisé par des ordres Basic spéciaux effectuant la gestion des tampons d'information par lesquels transitent les blocs envoyés d'une unité de traitement à l'autre. Des instructions telles que READ BUFFER ou WRITE BUFFER, assorties de tests, permettent de transférer à grande vitesse les informations d'une machine à l'autre.

— L'ordre d'écriture avec FORMAT, qui est l'ordre le plus élaboré d'utilisation de l'imprimante, peut en Basic III être utilisé également sur d'autres périphériques. Citons en particulier la possibilité d'écrire des informations complètement formattées sur disque, ce qui permet de réaliser des impressions ultérieures avec des programmes Basic très courts et standards ouvrant la possibilité de réaliser facilement du spool. L'ordre WRITE considérant l'écran comme un périphérique permet, comme nous l'avons déjà signalé, d'envoyer très commodément des masques élaborés sur l'écran.

Enfin, une instruction généralisée d'entrée/sortie IN/OUT permet de gérer les transmissions de données avec les possibilités de choix de parité que nous avons indiquées plus haut, la vitesse étant programmable dans l'instruction. Cette instruction peut bien sûr être utilisée à distance par ligne téléphonique avec modem, mais on peut aussi l'utiliser, bien entendu, en local. Dans ce dernier cas, citons deux applications principales : la première consiste à connecter des périphériques à interface série, tels que par exemple, des téléimprimeurs, des machines à dessiner, etc. Dans le second cas on pourra se connecter à un autre calculateur également muni d'un interface série. Il sera ainsi possible soit d'asservir l'autre calculateur en lui envoyant des chaînes de caractères de commande, soit de lire des enregistrements de fichiers que l'autre calculateur appellera les uns après les autres sur ses propres mémoires magnétiques (cette opération a été réalisée en particulier sur IBM 5100, WANG 2200, HP 9830 et HP 3000).

Nous venons donc d'énumérer les caractéristiques particulières du langage en ce qui concerne ses instructions. Un autre aspect est également important, c'est la façon dont le travail de programmation et de mise au point interactif à l'écran se présentera pour la personne qui va rédiger les programmes. L'un des avantages du Basic est d'être par nature un langage à mise au point interactive ; tout a été mis en œuvre ici pour que cette caractéristique soit poussée au maximum.

Mise au point interactive

Une analyse syntaxique immédiate de chaque ligne de programme frappé, est effectuée dès que la touche RETOUR CHARIOT est enfoncée. Si cette analyse syntaxique décèle une erreur, un message d'erreur avec numéro est immédiatement affiché derrière la ligne. On a alors la possibilité de retaper immédiatement un nouveau texte de ligne, ou encore si la ligne ne présente que peu d'erreurs, de manipuler le curseur, comme nous le verrons plus loin, pour les corriger. Dès que cette opération est terminée, lorsque l'on appuie à nouveau sur la touche RETOUR CHARIOT les messages d'erreurs sont effacés, et il ne reste plus que la nouvelle version de la ligne corrigée sur l'écran. On évite ainsi une accumulation de messages d'erreurs tout au long de l'écriture du programme original.

Contrôle complet du curseur permettant substitution, addition et effacement : les diverses touches de fonctions contrôlant le déplacement du curseur ont un double action. Outre l'action directe par frappe simple, si on laisse une touche en-

foncée plus d'une seconde environ, son action devient répétitive. Si par exemple on désire faire disparaître une série de caractères au milieu d'une instruction assez longue, par exemple un FORMAT d'édition, on positionne le curseur sur le premier des caractères à faire disparaître, puis, en appuyant sur la touche d'effacement, les caractères au bout d'une seconde commencent à être avalés à un rythme rapide, le reste de l'instruction se déplaçant au fur et à mesure pour venir combler le vide. De la même façon, on peut déplacer rapidement le curseur d'un point à un autre. Toutes les touches du clavier possèdent cette répétition automatique, ce qui permet par exemple de générer rapidement des chaînes alphabétiques ou numériques, par répétitions de caractères donnés.

Exploration rapide du listing du programme : lorsqu'on a demandé le mode de visualisation du programme, certaines touches du clavier prennent des valeurs de fonctions particulières. En appuyant sur la touche M, on fera monter à chaque fois le listing par demi-page, et en appuyant sur D on le fera descendre également par demi-page sur l'écran.

Appel de ligne : il suffit à un moment quelconque de taper un nombre correspondant au numéro d'une des lignes du programme et de frapper la touche RETOUR CHARIOT pour voir la ligne correspondante apparaître sur l'écran. Il est alors possible de modifier la ligne par manipulation du curseur ; dès qu'on redémarre l'exécution du programme, la ligne est insérée en bonne place dans sa nouvelle version. Cette manœuvre est particulièrement pratique lorsqu'en cours d'exécution pour mise au point du programme, on bloque subitement sur un message d'erreur, qui nécessitera un examen de la ligne qui vient de le provoquer.

L'exécution du programme peut se faire en pas à pas simple, ou avec répétition automatique, permettant de faire se dérouler le programme à faible vitesse. Des ordres tels que WAIT permettent de réaliser une pause momentanée pour examen de résultat ; STOP permet, lui, un arrêt du programme avec reprise ultérieure par la commande CONTINUE.

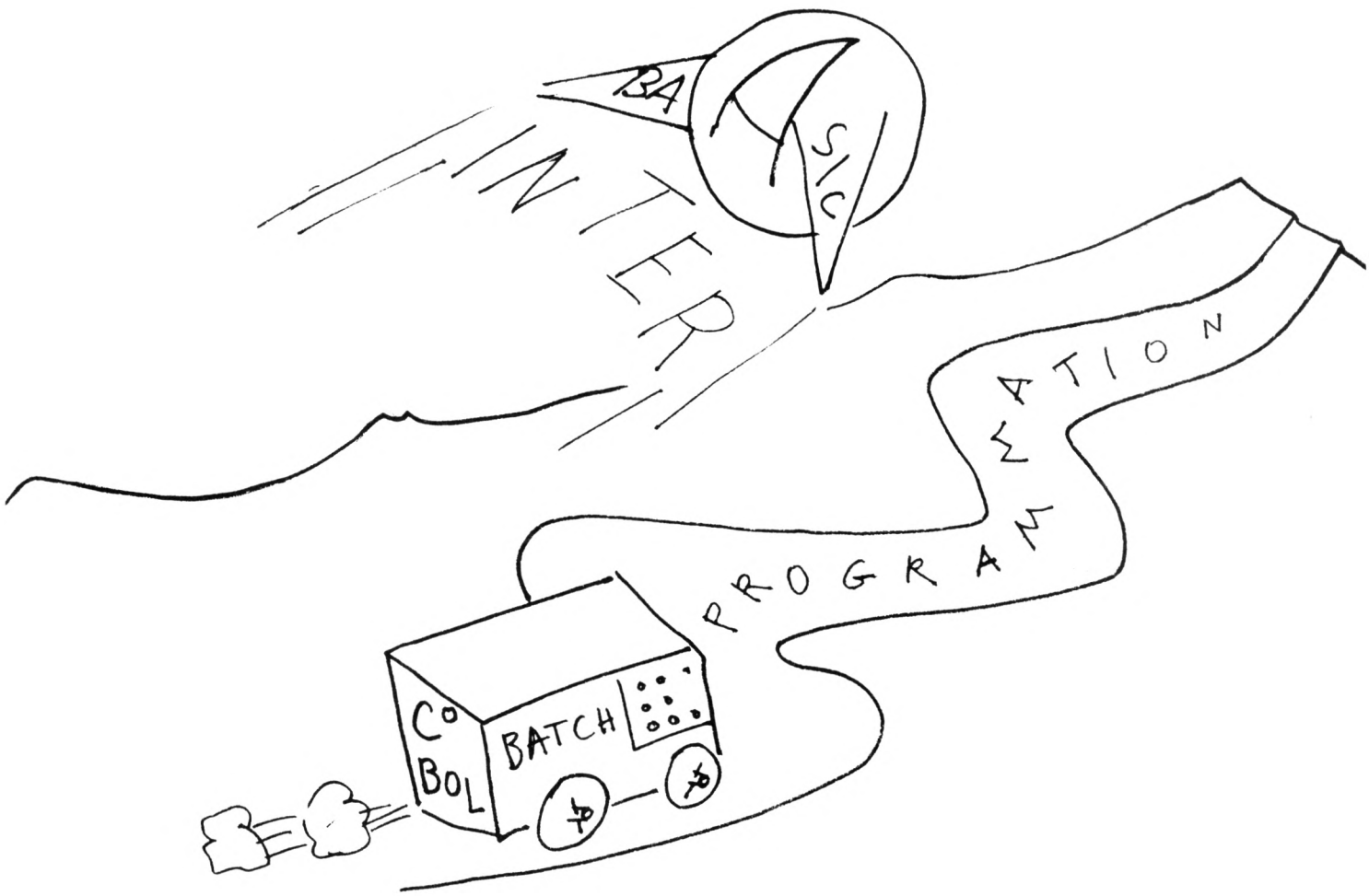
On peut à tout instant connaître l'état courant des variables utilisées dans le programme. Pour l'avoir au niveau d'une variable individuelle, il suffit de taper le nom de la variable au clavier, et la variable apparaîtra sur l'écran avec sa valeur courante. Une demande au clavier de l'affichage de la liste des variables permettra de connaître l'ensemble des variables utilisées par le programme avec indication de leur type. Lors d'un arrêt quelconque on peut modifier la valeur courante de la variable simplement en lui réassignant une nouvelle valeur, puis repartir en exécution du programme. Ces possibilités facilitent beaucoup la mise au point.

On peut enfin interrompre un programme en cours d'exécution, soit en exécution normale, soit un programme qui boucle à la suite d'une erreur, en pressant simultanément la touche SHIFT et la touche B pour provoquer un break. Il sera alors possible de modifier le programme, puis de repartir pour exécution.

Il est clair, à l'énoncé de toutes ces caractéristiques, que la programmation interactive telle qu'elle est permise par les structures de Basic modernes comme celle de Basic III, est beaucoup plus agréable et beaucoup plus rapide pour le programmeur que celle offerte par les compilateurs Cobol traditionnels utilisés dans un environnement de traitement par lots. Des reprises systématiques de chaînes de programmes écrites en Cobol et qui sont actuellement en cours de conversion en Basic III, montrent que le gain de temps amené par ces méthodes interactives dépasse tout ce que les prévisions les plus optimistes avaient pu laisser espérer.

Il ne faut pas oublier non plus que la rédaction minutieuse de la DATA-DIVISION du Cobol n'est pas nécessaire lorsqu'on travaille en Basic, ce qui explique également une partie des gains obtenus. Il faut avoir directement mis au point à l'écran un programme d'édition à format complexe, ou un calcul de paye tarabiscoté, pour se rendre compte du progrès accompli. Nous signalerons, pour en terminer avec ce point, qu'un centre de traitement qui a repris l'ensemble de ses chaînes Cobol non pas à partir des dossiers de programmation, mais des dossiers d'analyse, affirme avoir réalisé cette nouvelle programmation en quatre fois moins de temps que la première. Le constructeur n'oserait jamais annoncer de tels chiffres...

Le langage est disponible sous deux formes. La première est celle d'un interpréteur monobloc. La totalité de l'interpréteur réside donc en permanence dans la mémoire centrale de l'unité de traitement, que ce soit pour les fonctions de programmation ou d'exécution des programmes déjà écrits. Ces derniers n'existent que sous la forme de programmes source en clair. Dans la deuxième forme, le langage est toujours un interpréteur, mais il est scindé en deux parties, l'analyseur et l'exécuteur, dans le but de gagner de la place en mémoire. En cours de programmation, lorsqu'on désire effectuer des exécutions pour mise au point, puis corriger, l'analyseur et l'exécuteur seront alternativement appelés automatiquement en overlay en mémoire. Ceci bien évidemment dans un but de gain de place. Lorsque les programmes sont en exécution, seul l'exécuteur a besoin de résider en cours de traitement ; il est d'ailleurs possible d'exclure tout appel de l'analyseur, de façon à verrouiller les programmes pour empêcher des modifications ultérieures.



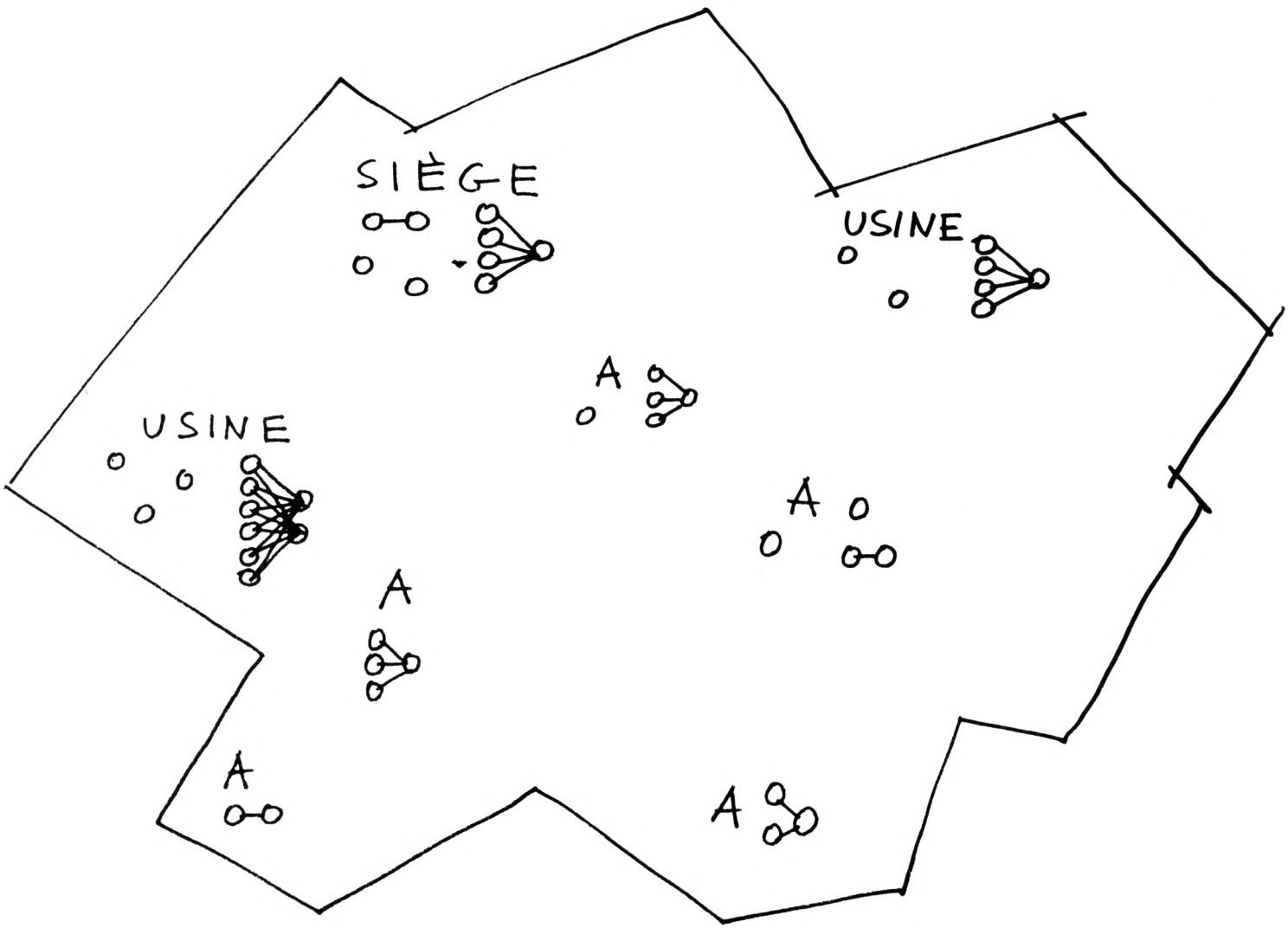
Enfin, signalons qu'un compilateur est actuellement en préparation pour le langage Basic III, ce qui permettra non seulement de gagner de la place, mais aussi d'accélérer l'exécution des programmes objet. Il sera donc possible d'écrire les programmes, de les mettre au point ou de les modifier en travaillant avec l'interpréteur en mode interactif, profitant ainsi de toutes les possibilités conversationnelles temps réel du Basic, et de ne compiler les programmes que comme opération finale, une fois qu'ils seront parfaitement mis au point.

L'informatique répartie

On voit donc qu'un seul produit hard/soft cohérent, l'unité "A", permet par agrégation progressive de couvrir une gamme importante de besoins en traitement de l'information. Déjà des versions plus rapides de microprocesseurs, où l'horloge deux mégahertz peut être remplacée par une horloge trois mégahertz, sont disponibles. La puissance dépasse cette fois le 360/30, ce qui offre une dimension supplémentaire de croissance dans les systèmes, pour les modules les plus chargés. La modularité complète des systèmes démarrant d'un système de base de configuration minimum pour aller jusqu'à des systèmes multipostes étendus, permet par ailleurs de démarrer petit, en vérifiant expérimentalement les concepts de l'informatique répartie, avant de s'engager à plus grande échelle.

L'informatique répartie commence donc à pouvoir vraiment se réaliser. Une hiérarchie souple d'ordinateurs peut être mise en place dans l'entreprise. Les communications entre eux se feront, selon l'urgence, soit par envoi postal de diskettes, soit par transmission de données. Le coût d'une carte V 24 est de 1 000 F environ et celui d'un modem de l'ordre de 2 000 F. Si en un point géographique donné la puissance devient insuffisante, on ajoute d'autres unités "A", soit en version autonome, soit interconnectées.

Les informations saisies et traitées sur place, donneront lieu à la création d'informations résumées qui remonteront vers le siège pour consoli-



dation. Selon le genre d'organisation adopté, il peut y avoir ou non un gros ordinateur au siège. Vraisemblablement son rôle est surtout de réaliser une synthèse générale des résultats, et de préparer des tableaux de bord, les traitements et l'impression étant effectués en majeure partie localement. L'ordinateur du siège sera sans doute connecté à un grand centre de temps partagé, pour des travaux importants, mais occasionnels, de recherche opérationnelle et de prévisions.

Que devient dans tout ceci le responsable informatique ? Notre avis est qu'il se portera mieux, et aura une situation souvent moins périlleuse dans l'entreprise que ce n'est le cas actuellement. Brodons un peu sur ce thème.

«*... Ça y est. Le système d'ordinateur hiérarchisé a fini par se mettre en place dans l'entreprise par petites étapes, avec d'ailleurs un accueil plutôt enthousiaste de la part des utilisateurs. Le directeur de l'informatique est délivré des insomnies causées par la montée des coûts, la crainte des grèves, de l'incendie et la rareté des bons ingénieurs système. Ses analystes et ses programmeurs sont égaillés dans la nature en train d'aider et de former les utilisateurs locaux de plus de 200 unités "A". Ayant enfin des loisirs, il vient de réaliser un modèle d'entreprise "pour le plaisir", alors qu'il n'avait plus programmé une ligne depuis dix ans, modèle qui fait maintenant la joie du président*».

8 - 3. CONCEPTION MODERNE DES ORDINATEURS MOYENS

A mi-chemin entre les ordinateurs géants, qui - comme nous l'avons vu - ne profitent guère des progrès des circuits intégrés, et les micro-ordinateurs de plus en plus évolués aux architectures diversifiées, que deviennent donc les ordinateurs moyens ? Ont-ils réussi à franchir le mur du son de l'électronique ?

En fait, ils ont tourné autour. Devant la quasi impossibilité de définir des sous-ensembles de logique correspondant à des fonctions d'usage général suffisamment répétitive, c'est une autre solution qui a été cherchée, et trouvée.

L'un des premiers à l'avoir mise en œuvre industriellement est Gene Amdahl, dans sa fameuse série d'unités centrales "compatibles", les Amdahl 470, qui peuvent prendre en souplesse la relève d'une unité centrale de 370 IBM en en simulant si exactement le fonctionnement que les programmes de l'utilisateur n'ont pas à subir un iota de modification pour pouvoir tourner...

Mais le grand géant d'Armonk n'est finalement pas resté à la traîne : lui que beaucoup de concurrents avaient accusé de se reposer sur ses lauriers en continuant à produire sans changement de fond la série 370 déjà vieille de neuf ans en 1979, vient dans un grand mouvement de se positionner en rang convenable dans la technologie des circuits intégrés. Les nouveautés dévoilées nous paraissent définir une orientation caractéristique pour la décennie 80-90. Pendant ce temps-là, d'autres constructeurs de compatibles progressent rapidement : National (eh oui, un fabricant de composants) avec son système/400 à architecture à microprocesseurs fonctionnels répartis (débit identique au 370/145), Two Pi avec son V 32, Magnuson Systems avec son M 80 à architecture multiprocesseurs redondants utilisant le vote triple à majorité.

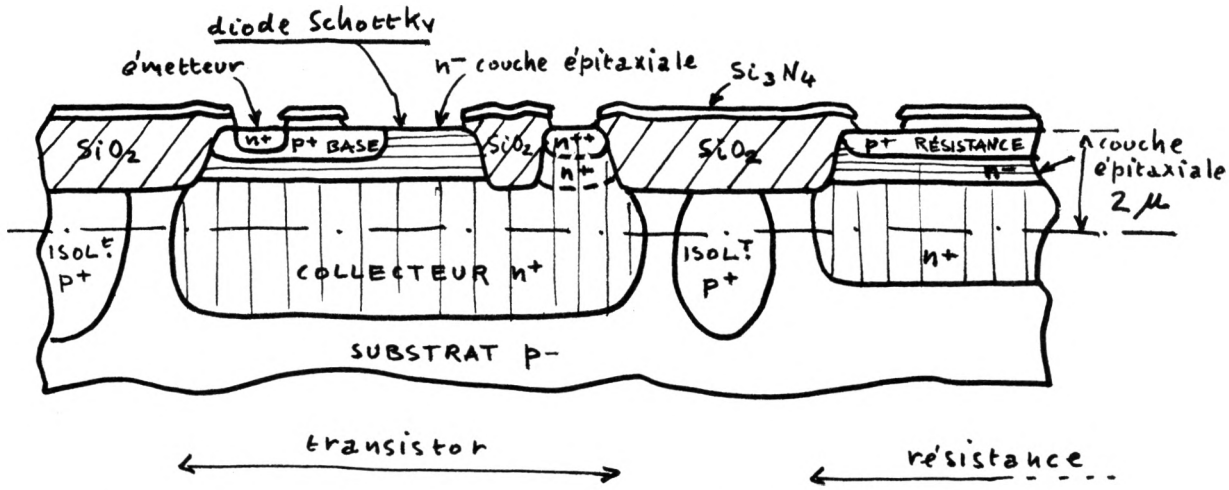
8 - 3 - 1. Les 704 portes d'IBM : à discrétion

704 est sans doute un nombre magique chez IBM. C'était le nom du premier vraiment gros ordinateur de la Compagnie, c'est aujourd'hui le symbole de son nouveau type de confection : après le sur mesure, la mesure industrielle.

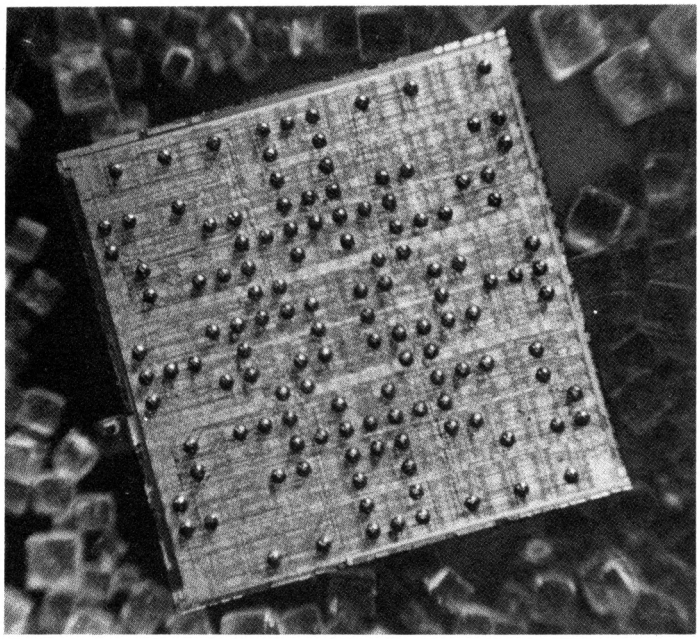
Notre élément de base est la *tranche maîtresse* (master slice), une puce carrée de 4,5 mm de côté qui contient 704 portes réalisées en technologie TTL Schottky (délai de propagation 3 à 5 nanosecondes). Elle est en particulier la base du Système/38, machine de bonne puissance dont l'unité arithmétique et logique comprend 20 000 portes.

L'ensemble de ce processeur tient sur une carte de 25 cm sur 37 cm portant 29 circuits intégrés, tous identiques... à l'exception de la liaison des portes entre elles sur chaque puce. Deux niveaux d'inter-connexion métalliques réalisés à la demande en alliage aluminium-cuivre relient les portes les unes aux autres, tout à fait (à l'échelle microscopique) comme les pattes des composants sont reliées les unes aux autres par les pistes d'un circuit imprimé double face. Ici le rôle des trous métallisés est joué par des passages (des *vias*) à travers les couches isolantes de silice.

Un troisième niveau de métallisation (le plus éloigné du silicium) est normalisé comme disposition, et débouche sur 132 points de connexions externes disposés en quadrillage sur toute la surface de la puce, et non pas seulement sur les bords. Après déposition d'une couche protectrice de chrome-cuivre-or, chacun de ces points reçoit un petit globule de soudure étain-plomb. La puce est ensuite retournée et posée sur un micromodule céramique porte-puce carré, de 25 mm de côté environ, à 116 broches en quadrillage. Un réseau métallisé est bien entendu déjà déposé sur la céramique ; on passe le tout au four, les globules de soudure fondent... et avec un peu de chance tout est bien soudé ! En faisant le compte sur vos doigts, vous avez sans doute déjà compris que plusieurs des points soudés de la puce peu-

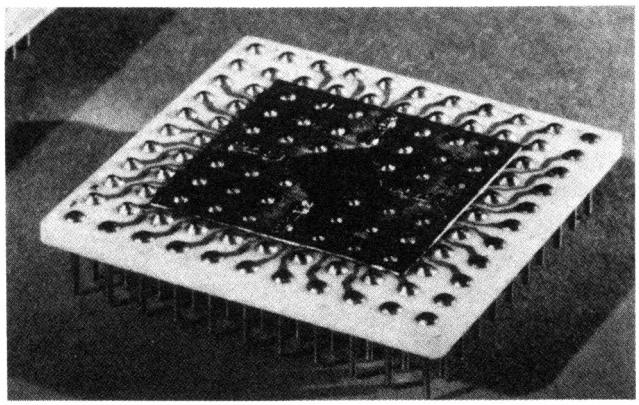


Structure complexe de la tranche maîtresse en TTL Schottky



La puce avec ses globules de soudure.

vent être reliés à une même broche, en particulier pour distribuer les tensions d'alimentation (en fait il y a 94 ports de signaux sur la puce).



Le micromodule céramique porte-puce a deux niveaux d'interconnexion.

Pour en revenir à notre processeur complet, sur les 29 circuits intégrés... il y a 22 types différents ! Ceci est une parfaite illustration des problèmes décrits dans "La confusion" et "Le mur du son", et de la méthode utilisée pour les résoudre. Cette dernière n'est économiquement valable qu'au prix d'une normalisation à un niveau intermédiaire, que nous allons voir de plus près, et d'un outil logiciel important.

Les éléments de base diffusés sur la puce comprennent des transistors de 3 microns sur 8, accompagnés de leur diode shunt Shottky de 5 microns sur 6 (en shuntant l'espace collecteur-base du transistor elle limite sa saturation : le nombre de porteurs à neutraliser lors de la commutation étant dès lors plus faible, la commutation s'effectue plus rapidement) et de résistances de 4,5 microns sur 70. Il y a en plus 60 circuits drivers, alimentés en — 4,5 V (la logique est alimentée en — 1,5 V).

8 - 3 - 2. Une bibliothèque métallique

Les gabarits d'interconnexion définissant un type particulier de porte (il y en a une centaine de différentes) appelés chacun un *livre* (book), sont rangés dans la bibliothèque sur disques du système de conception assistée par ordinateur, sous une forme numérisée repérant les contours des pistes métalliques, etc.

Le logiciel de CAO, incorporant toutes les règles de câblage, choisit les livres et résistances adéquates selon le cas de figure, calcule les chutes de tension dans le réseau et, si elles sont trop fortes, élargit les pistes métalliques, puis incorpore les drivers pour les sorties, d'après le diagramme logique de la puce qu'il a déjà vérifié par simulation. Il effectue une optimisation d'ensemble pour raccourcir les longueurs de conducteurs, sous le contrôle des ingénieurs. En effet, si le calcul des charges et des délais de propagation peut être automatisé, il faut vérifier que le processus d'optimisation dans le cours de sa recherche brutale d'efficacité n'attribue pas par malchance des longueurs physiques très différentes à des trajets de signaux en concordance, risquant de faire ainsi disparaître l'un des avantages que l'on est en droit d'attendre de la miniaturisation de l'électronique !

Les délais de propagation dans les interconnexions sur une puce sont parfois non négligeables. Dans une mémoire dynamique de 16 K par exemple, une piste de silicium polycristallin de 2 millimètres de long introduit un délai de 15 nanosecondes, alors que si elle est en molybdène le délai n'est plus que de 0,2 ns. Si la taille de la puce croît, les délais évoluent vite : une piste de 4 mm correspondra avec les deux mêmes matériaux à 60 ns et 1 ns !

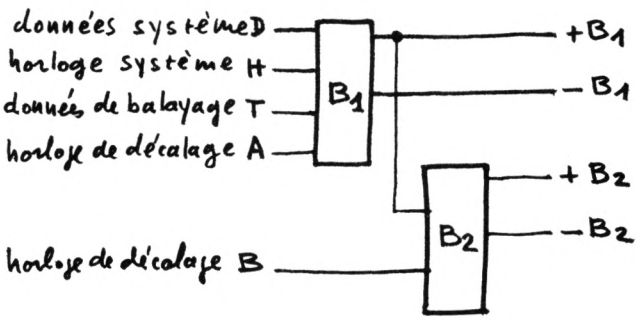
Une fois tout vérifié, le système de CAO produit les informations permettant de piloter un appareil à faisceau d'électrons qui impressionnera directement une couche de résine photosensible déposée sur la puce : on peut ainsi obtenir très rapidement des prototypes. Pour une fabrication en série, on pourra avoir, bien sûr, recours aux techniques de masquages conventionnelles que nous connaissons bien.

8 - 3 - 3. Conception structurée pour tests automatiques

Lorsque l'Amdahl 470 est apparu il y a quelques années, utilisant une méthode ressemblant à celle que nous décrivons ici, quoiqu'avec une technologie différente, on se rappelle le remarquable système de télé-diagnostic qu'il inaugurait. En cas de problème, tout en restant au site central de maintenance relié par lignes téléphoniques aux ordinateurs de clients, les ingénieurs pouvaient consulter, afficher sur écran, traiter par programmes de diagnostic, l'état de dizaine de milliers de bascules de l'ordinateur soupçonné ! Plus l'électronique devient dense, plus il est nécessaire de disposer de moyens de test puissants : nous allons voir comment IBM a abordé la question sur le Système/38.

Le gros problème amené par l'augmentation d'intégration, nous l'avons déjà signalé pour la boîte noire des microprocesseurs - fort heureusement standard, peu coûteux et donc faciles à remplacer, mais surtout très fiables - est que les signaux internes ne sont plus accessibles.
Par ailleurs, la complexité de la logique devient telle que si l'on veut continuer les anciennes méthodes de test en alimentant d'impressionnantes séries de signaux destinés à mettre dans tous leurs états les circuits internes, les temps deviennent très longs.

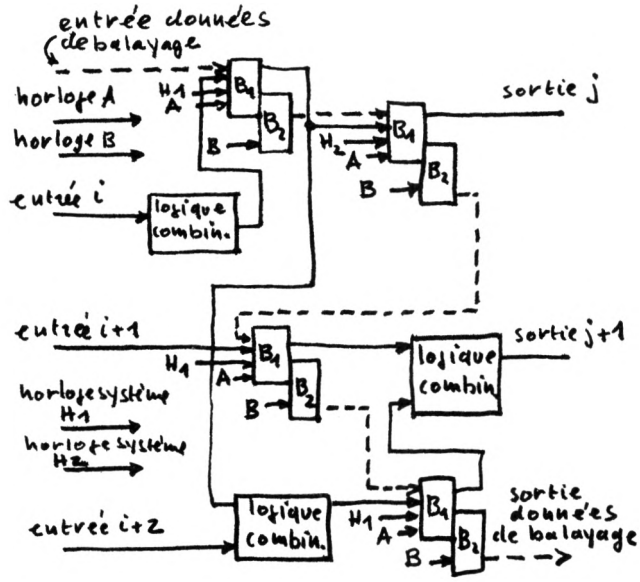
Le système employé ici est le LSSD (Level Sensitive Scan Design, conception à balayage détecteur de niveaux). Le seul élément de mémorisation autre que des RAM permis dans la conception logique est la bascule à verrouillage, étape d'un registre à décalage (SRL, shift register latch), un SRL est composé de deux bascules de type D (revoyez le paragraphe 2 - 2 - 7 si vous avez un peu oublié bascules et registres à décalage) interconnectées comme l'indique le dessin, de façon à former *un* étage tout à fait particulier de registre à décalage. La première bascule (B1) sert à la fois au



fonctionnement de la logique et aux tests, alors que B2 n'est là que pour faciliter les tests (bien qu'on puisse l'utiliser à d'autres usages).

B2 n'a qu'une seule entrée de données, connectée à une des sorties de son compagnon B1, et une entrée horloge B permettant de mettre B2 à l'état de la sortie de B1 connectée (c'est une sortie normale, B2 sur réception de l'horloge prendra donc l'état de B1).

B1 par contre peut être positionnée à 1 à partir de deux sources, à l'aide de deux horloges, H et A. Son entrée *données de balayage* marquée T (test) est reliée à l'une des sorties d'une bascule amont de type B2 : c'est sur réception de l'horloge de décalage A que se chargera l'entrée T (l'entrée D est commandée par l'une des deux horloges système). On voit déjà la dualité de rôle de B1, à la fois élément de mémorisation dans le réseau logique, et élément de test.



De proche en proche, comme l'indique le dessin, toutes les bascules sont reliées entre elles, formant une longue chaîne qui serpente sur la puce : la sortie de la première B2 est connectée à l'entrée de la B1 suivante, etc., jusqu'à la dernière. L'entrée T de la toute première B1 est reliée à une broche du porteur céramique de la puce permettant l'entrée pour test (TE), et la sortie de la toute dernière B2 à une deuxième broche de sortie test (TS). De même, les entrées horloges A et B de toutes les paires de bascules B1/B2 sont connectées aux broches A et B. A partir de ces connexions externes, nous allons pouvoir mener les tests.

Il nous semble qu'il y a ici un intéressant parallèle à tracer avec la programmation structurée. Dans cette dernière, le fait de restreindre considérablement sa liberté par l'unique recours à un petit nombre de primitives simples (alternative, répétitive, etc.) permet de clarifier les programmes, de rendre beaucoup plus faciles les tests, mais

augmente le nombre des instructions. Or, un programme, c'est de la logique, à l'état encore plus pur si l'on ose dire ! Le fait de restreindre sa liberté à un seul type d'élément de mémorisation permet de faciliter les tests, au prix d'une augmentation du nombre des éléments logiques : cette dernière se chiffre à un peu moins de 20 %.

8 - 3 - 4. Le grand pavois

Tout est prêt maintenant pour tester. On commence par envoyer une longue chaîne de zéros et de uns sur la broche TE, la mettant ainsi en place pour initialiser les bascules après que les décalages successifs à chaque impulsion des horloges A et B se soient effectués. Appliquons maintenant une combinaison d'états logiques aux broches d'entrée de la puce : sa logique combinatoire va prendre un certain état. Une partie du réseau logique se trouve entre les sorties des bascules (déjà initialisées) et les broches de sortie : on dispose là d'un premier ensemble d'états à analyser pour savoir si ladite logique fonctionne bien. Reste la partie de la logique entre les broches d'entrée et les entrées des bascules. On cycle une fois les horloges système H1 et H2, et les états de cette logique sont piégés dans les bascules B1. Puis on remet en route les horloges A et B pour décaler les états qui sortent en série sur TS, également prêts à être analysés.

On voit donc qu'on dispose de ce qui manquait cruellement : des points de tests disséminés en plein milieu de la puce. On pourrait imaginer que les LEDs y soient connectés, si la puce avait une surface 10 000 fois plus grands ! Après avoir chargé le pattern (motif? si vous préférez ?) binaire d'essai, on récupère l'état des lampes de test en tirant une longue guirlande de lampes allumées ou éteintes, tel le grand pavois de nuit d'un bâtiment de guerre un peu négligé où l'électricien de bord ne remplacerait pas les ampoules grillées...

Et cette guirlande peut être très longue : car en effet, vous vous en doutez probablement déjà, on pourrait relier ensemble la broche TS d'une puce... à la broche TE de la suivante, créant une chaîne de milliers de bits en zig-zag sur toute notre carte processeur ! Là nous allons un peu loin : pour diminuer le temps de chargement de patterns, on crée en fait des chaînes un peu plus courtes en regroupant les puces, de façon à charger en parallèle, la carte étant enfichée dans le matériel de test au service contrôle de qualité.

Le test si la carte est montée dans l'ordinateur est plus difficile, les états des sorties logiques ne pouvant être observés, puisqu'elles sont maintenant reliées à la mémoire, aux canaux, etc. Par contre, l'état des bascules est toujours accessible, permettant un test déjà valable.

Naturellement, cette technique ne permet de tester que les défauts apparaissant en fonctionnement statique. La comparaison, au banc de test, des patterns relevés avec les patterns créés par les programmes de simulation permet de tester plus de 98 % des défauts statiques. Sur le terrain, on ajoute le passage de programmes de diagnostic de type classique pour compléter par des tests dynamiques.

On peut étendre le concept au niveau système. Un SCA (system control adapter), muni de son propre microprocesseur pilote momentanément H1 et H2 : il les arrête, lit le pattern de test sur un fichier, le charge dans le processeur, cycle H1 et H2, retire le pattern résultat et le compare avec les résultats normaux lus sur le fichier. Le SCA gère également le pupitre de maintenance, permet d'altérer manuellement l'état des bascules, et le visualise sur écran.

A quand le basculement automatique d'éléments de secours par le SCA, pour auto-dépannage, introduisant dans l'environnement informatique de tous les jours les techniques, jusque-là réservées à l'espace, du projet STAR (self-testing and repair) ?

8 - 3 - 5. L'Univac 1100/60 : un gros ordinateur entièrement réalisé avec des microprocesseurs

Dernier né de l'immortelle série 1100 qui a fait (et fait) le succès d'Univac dans les couches de gros utilisateurs scientifiques et de gestion (ayant souvent des réseaux de transmissions étendus) le modèle 60 est un exemple unique qui pourrait bien être annonciateur d'une nouvelle tendance en matière d'architecture d'unités centrales puissantes.

IBM a choisi une voie : la puce "bonne à tout-faire" personnalisée par métallisation, comme brique élémentaire (déjà complexe) de construction. Cette approche continue la verticalisation commencée vers 1963 avec le Système/360 : IBM fabrique une grande partie de ses propres composants, avec tous les avantages et les inconvénients que cette solution comporte (il est quand même difficile d'être simultanément à la pointe partout).

Univac vient de choisir une voie diamétralement opposée : l'adoption d'un microprocesseur standard (vous avez bien lu !) commercialement disponible. Il s'agit du remarquable microprocesseur en tranches Motorola 10800, en technologie ultra-rapide ECL. Le 1100/60 ne comporte pas moins de 68 de ces tranches, montées sur 72 cartes à 10 couches dans le plus gros modèle (fond de panier à 16 couches).

La réduction de coût due à l'utilisation de composants standards est particulièrement sensible sur cette unité centrale à mots de 36 bits dont la logique générale bien pensée n'a pas bougé depuis le début des 1100 : elle existe en six ni-

veaux de puissance croissante, et l'on peut passer de l'un à l'autre par transformation sur le site, ce qui est exceptionnel.

Le processeur est un ensemble bicéphale comprenant deux sous-processeurs, composé chacun de deux groupes de neuf tranches du 10800 (larges de quatre bits chacune, d'où nos 36 bits). Ces groupes fonctionnent en pipe-line avec recouvrement d'un maximum de quatre micro-instructions (stockage d'un résultat/exécution/transfert de l'opérande/génération de l'adresse de la prochaine instruction).

Les deux sous-processeurs travaillent exactement en parallèle en synchronisme, avec comparaison des résultats : si une discordance apparaît, l'exécution de l'instruction est automatiquement répétée, ce qui permet de récupérer les erreurs transitoires sans avoir à provoquer une interruption type "machine check", prenant beaucoup plus de temps. Si après répétition il y a toujours discordance, une panne est signalée au pupitreur. Ce principe augmente considérablement la fiabilité.

Le télé-diagnostic est ici aussi la règle, avec chargement à distance de registres dans les microprocesseurs, exécution et comparaison des résultats. L'importante quantité de logique par puce du système 10800 permet d'isoler facilement le boîtier fautif.

La mémoire centrale est en boîtiers dynamiques 16 K à 575 nanosecondes. Sa taille va de 512 K à 1024 K mots de 36 bits (2,4 à 4,9 Mo) par incréments de 256 K mots.

On peut considérer que cette série est une réponse à la série 4300 d'IBM. Mais les 1100/60 montent plus haut. Le plus petit modèle, le C1, équivaut au 4331. Par adjonctions successives d'instructions étendues, de mémoire cache, puis par couplage de processeurs, on aboutit au H2, cinq fois plus performant, et qui est du niveau du 3032 IBM : c'est un haut de gamme.

8 - 4. LE SUPER ORDINATEUR SUPRACONDUCTEUR

Avant de quitter le domaine des gros ordinateurs, nous allons dire quelques mots d'une des formules d'avenir possible pour un ordinateur géant... par les possibilités, mais comme on va le voir, pas par la taille !

Vous l'avez vu d'après le titre, il fonctionnera dans l'hélium liquide, à une température si basse que certains métaux et alliages voient leurs propriétés électriques changer du tout au tout : leur résistance électrique devient absolument nulle ! C'est la supraconductivité, due au fait que les électrons circulent en se suivant deux par deux, le second comme un canot tiré sur la vague de sillage arrière d'un bateau dans une position (où il ne faut jamais le mettre en pratique !) telle qu'il dévale perpétuellement la vague vers le bas... Les deux systèmes de vagues s'annulant, il n'y a plus de résistance de sillage.

Un courant lancé dans une boucle fermée y circulera indéfiniment, ce qui bien sûr permettra de réaliser une cellule de mémoire. Mais pour la logique il nous faut un élément équivalent au transistor, ou plus précisément à la jonction entre semiconducteurs : il existe, c'est la jonction Josephson.

8 - 4 - 1. La jonction Josephson mise en pratique

Rappelons le principe de la jonction Josephson : elle comprend deux bandes supraconductrices séparées par une couche extrêmement mince d'isolant. Par effet tunnel (mécanique quantique) les électrons traversent cette couche sans éprouver aucune résistance, et la tension entre les deux faces de la jonction est donc nulle : c'est l'état zéro. Si on applique un champ magnétique (pas trop fort, pour ne pas faire disparaître la supraconductivité dans les bandes), la jonction se met à présenter une résistance, bien que l'effet tunnel continue. Le passage du courant fait alors apparaître une tension entre les faces de la jonction : c'est l'état un.

Depuis longtemps ce type de jonction est expérimenté, car il présente des temps de commutation remarquables.

Mi-79, Tushar Gheewala, chercheur du centre IBM Thomas D. Watson, a inventé et expérimenté un circuit logique à effet Josephson d'un type nouveau. C'est actuellement le circuit au temps de commutation le plus court jamais réalisé : 13 picosecondes (vous avez bien lu, 13 millièmes de nanoseconde) pour une porte OU et 26 pour une porte ET. Ces temps sont trois fois plus courts que ceux des précédents circuits à effet josephson, et dix fois plus courts que ceux des circuits à transistors ultra-rapides. Le temps de commutation proprement dit de la porte OU est de 7 picosecondes, et le temps de parcours du signal jusqu'à la porte suivante de 6 picosecondes (à la vitesse de 70 microns par picoseconde dans le conducteur, qui est un sandwich d'alliages métalliques et d'isolants).

Rappelons que le nom de ces phénomènes de commutation supraconductive (nécessitant une température proche du zéro absolu, soit moins 273°C) vient de celui du physicien britannique qui les avait théoriquement prévus, Brian Josephson.

Ces nouveaux circuits sont appelés "logique à injection de courant", et leur élément essentiel est une porte à injection constituée de deux jonctions Josephson reliées par une inductance :

cette porte effectue directement une logique ET. Permettant des courants plus intenses, elle a une meilleure entrance et une meilleure sortance (de 2 à 6 ou 8). Pour une porte ET à 4 entrées et 2 sorties, le délai total est de 41 ps ; il augmente de 6 ps par sortie additionnelle. La consommation est de 6 microwatts en moyenne.

Ces performances ont été obtenues avec une largeur de lignes de 2,5 microns. Une largeur de 1 micron (par lithogravure à faisceau d'électrons) permettrait de gagner encore, par raccourcissement proportionnel du temps de commutation, mais surtout par le fait que les portes se rapprocheraient les unes des autres, réduisant le temps de parcours du signal...

8 - 4 - 2. De Josephson à Zardoz

A partir de ces résultats, on peut espérer réaliser un ordinateur ultra-rapide, ce qui lui imposera d'ailleurs une taille très réduite. IBM travaille actuellement sur un projet (à long terme) dans lequel le trajet le plus long à l'intérieur du processeur serait de 7,5 cm, et entre processeur et mémoire centrale de 30 cm. Ces contraintes imposeront un encombrement total (processeur plus mémoire) de 8 cm sur 8 cm sur 10 cm, soit un peu moins que le volume intérieur d'une bouteille de Bordeaux !

Cette taille très réduite est évidemment absolument nécessaire pour réduire sufisamment les temps de propagation des signaux afin d'atteindre l'extrême rapidité souhaitée. Le cycle de base serait en effet de quatre nanosecondes (soit 250 MHz !) contre plus de 50 pour les plus grosses machines actuelles. L'antémémoire (ou mémoire cache) permettant l'anticipation ferait 256 Ko avec un temps d'accès de une nanoseconde, et la mémoire centrale 64 millions d'octets avec un temps d'accès de 15 nanosecondes, ce qui est décidément bien lent... mais elle ne consomme rien au repos, puisqu'elle est à boucles supraconductrices.

Très curieusement, sur le plan de la fabrication, bien qu'il ne soit pas supraconducteur, on ferait appel à des puces de silicium, tout simplement parce que la micrométallurgie/photolithogravure du silicium a maintenant 20 ans d'expérience, et est arrivée à une extraordinaire précision (que vous verrez encore plus en détail dans le chapitre qui suit).

Le silicium aurait donc principalement ici, un rôle de support pour les pistes en alliage plomb, iridium et or. On est déjà arrivé à fabriquer 1000 jonctions Josephson par puce. C'est un début, mais il faudra aller beaucoup plus loin. Les minuscules cartes seraient elles-mêmes en silicium, pour l'homogénéité du coefficient de dilatation (n'oublions pas qu'on aura à passer de temps en temps de la température ambiante à celle de l'hélium liquide), les "soudures" sur le fond de panier étant réalisées par des gouttes de mercure qui se solidifient dans l'hélium ! Il y a sûrement encore pas mal de travaux pratiques à mener assidûment.

L'un des points fondamentaux sous-jacents, qui font le gros intérêt des technologies évoquées ici, est la très faible dissipation thermique. Un ordinateur ultra-rapide, ultra-compact, à circuits intégrés, gros comme un pamplemousse, fondrait dès qu'on le mettrait sous tension. Un deuxième avantage est que les pistes de connexion étant à résistance nulle, la propagation des signaux y est nettement plus rapide. Par contre, la contrainte du bain d'hélium liquide n'est pas mince...

C'est ici qu'on peut juger à quel point des appellations telles que "gros ordinateur" ou même " ordinateur lourd " deviennent périmées ! Pensons plutôt à l'avenir comme l'a justement fait John Boorman dans Zardoz. Le tout puissant ordinateur qui contrôle le Vortex doit son invulnérabilité à son extrême petitesse : personne ne sait où il est !

Un jour nous aurons peut-être des composés complexes cristallins supraconducteurs à la température ordinaire (des recherches sont en cours depuis longtemps), pouvant émettre facilement des pinceaux lasers par endroits pour communiquer, comme le font déjà les diodes à injection à l'arséniure de gallium ou aux grenats d'yttrium et la scène capitale de Zardoz deviendra réalité. Sean Connery l'Exterminateur, entré par effraction dans le calme protégé du Vortex, finit par s'affronter à l'ordinateur qu'il a enfin découvert : c'est un petit cristal aux mille facettes rayonnant des faisceaux lumineux multicolores échangeant information et énergie, lévitant dans le champ magnétique terrestre et doté de mobilité, sans doute grâce à ses boucles internes supraconductrices de sustentation...

CHAPITRE 9
LES NOUVEAUX MICROS

Après cette incursion chez les "concurrents", les ordinateurs lourds, nous allons revenir, pour y rester désormais, dans le domaine des micros.

Domaine foisonnant s'il en fût. La densité des composants par millimètre carré de puce doublant en moyenne à peu près tous les 18 à 24 mois... les nouveautés se succèdent. En cette année 1979, on ne parle plus que de microprocesseurs à 16 bits. Est-ce à dire que les 8 bits sont morts ? Nous ne le pensons pas. Du moins, pas encore. Mais les 16 bits sont déjà une réalité avec laquelle il faut compter, avec laquelle noustravaillons du reste chaque jour. Nous vous proposons de commencer notre chapitre par une redécouverte - mais oui - celle des 16 bits... qui existent depuis bien longtemps ! Et de la prolonger par une découverte, celle des nouveaux venus.

9 - 1 LES DESTINS DES MICROS : UN DIORAMA EN 16

Pourquoi avons-nous préféré diorama à panorama, qui paraîtrait plus naturel ? Eh bien... parce que dans le diorama le spectateur est dans l'obscurité, comme devant cette multiplicité de micros vieux ou neufs, d'architectures et de caractéristiques si différentes.

Le PACE

Le tout premier fabricant de composants ayant réussi à produire un micro à 16 bits sur une seule puce - honneur à lui - est National Semiconducteur Corp., familièrement NS. L'engin s'appelle le PACE, et possède une architecture assez voisine de celle d'une famille de minis bien connue, la famille NOVA de Data General. Il est assez séduisant probablement pour un fabricant de se dire que l'architecture qu'il adopte a été mise à l'épreuve sur le terrain. NS n'a pas été le seul, tant s'en faut, à le penser, comme nous le verrons.
Malheureusement, la technologie a trahi le projet. La puce étant réalisée en MOS canal P, donc lent, n'offrait pas d'avantage de rapidité évident par rapport aux 8 bits les plus populaires en MOS canal N. Son succès a été faible. Sans doute est-ce à cause de ce départ initial assez lent que la version suivante en canal N, appelée INS 8900 n'a pas bien pris non plus.

Le microNOVA mN 601

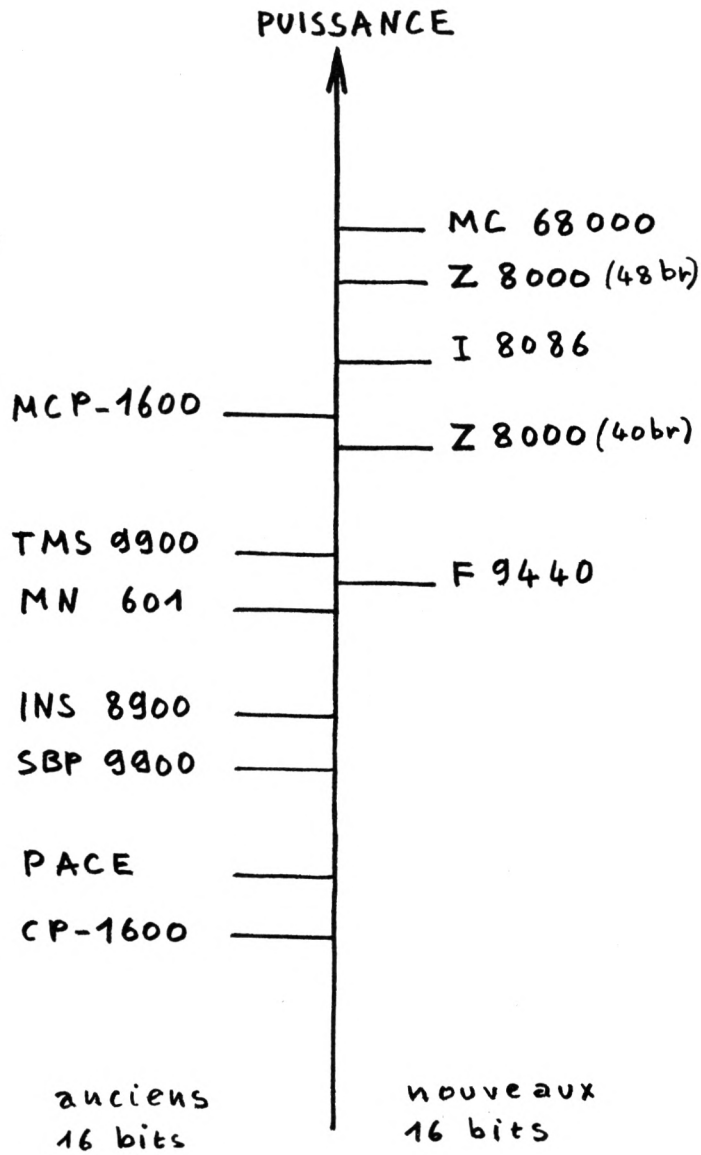
De son côté, Data General n'a sans doute pas voulu, en tant que constructeur de minis, rester avec un flanc sans défense face aux micros qui montaient (de même que Digital), et a lui aussi conçu un micro s'appuyant sur l'architecture NOVA : c'est le MicroNOVA mN 601. Sa force et sa faiblesse ont la même origine : tout a été mis en œuvre pour assurer une bonne compatibilité avec le jeu d'instructions NOVA, à fin de profiter au mieux de l'importante quantité de logiciel réalisée sur ces minis depuis pas mal d'années. Ce qui est attractif.

Le revers de la médaille est que cet objectif étant prioritaire par rapport à d'autres, la puce obtenue demandait 4 tensions d'alimentation différentes, était assez lente (près de 3 microsecondes pour les instructions les plus rapides), et surtout nécessitait un important entourage TTL et autre pour être utilisable. De plus, le fabricant ne poussait pas très violemment au début la vente du boîtier microprocesseurs, semblant préférer la vente de cartes complètes... et peut-être même, qui sait, de systèmes entiers. Ce qui est sûr, c'est que la percée sur le marché a été relativement faible.

Les TI 9900

Le cas de Texas Instruments est assez différent : nous avons affaire ici avant tout à un fabricant de composants, construisant aussi des minis, la série 990. TI a conçu son premier microprocesseur 9900, de même que DG, comme un prolongement vers le bas de sa gamme de minis, mais avec ici une nette intention de vendre le boîtier aussi bien seul que sur cartes ou dans un système complet. Il y a maintenant plusieurs types de 9900, le plus connu étant le TMS 9900 en MOS canal N, bien placé en puissance comme on le voit sur le croquis récapitulatif. Les micros 9900 présentent une caractéristique tout à fait particulière : ils ont des instructions à deux adresses, travaillant directement en mémoire. Par exemple, pour déplacer un bloc de données en mémoire, il n'est pas nécessaire de charger et de décharger un registre dans l'unité arithmétique et logique, comme c'est le cas avec des instructions à une adresse. Il n'y a d'ailleurs pas de registres sur la puce : ils sont en mémoire, mais utilisent un adressage raccourci... sinon les instructions de registre à registre feraient une quarantaine de bits de longueur !

Là encore, force est de dire que le 9900 n'a pas eu un très grand succès. La nouveauté de sa structure est très certainement un peu déroutante (tout au moins pour ceux qui n'ont jamais programmé un 1401 ou un Gamma 30 !), et le fait d'avoir les registres en mémoire peut faire craindre une certaine lenteur. En réalité, l'existence des instructions à deux adresses fait gagner du temps par ailleurs, et le bilan global est correct. Peut-être un autre point d'inquiétude a-t-il été le boîtier très encombrant à 64 pattes ?



Enfin, l'un des boîtiers essentiels de support, le CRU (communications register unit) est une espèce de super 8255 : quant on a vu (chapitre 7-9-2) l'usine à gaz qu'est déjà ce dernier, on peut supposer que la complexité du CRU a un peu effrayé les nouveaux venus au microprocesseur...

La présence d'une bonne quantité de logiciel de base aurait dû être un argument supplémentaire pour le 9900, mais il semble que ce facteur ait assez peu joué. Bref, le 9900 n'a certainement pas eu le succès mérité.

Le MCP-1600

Voici le dernier de nos 16 bits anciens... qui nous fournira une transition vers les nouveaux comme nous allons le voir.

Le MCP-1600 est lui aussi lié de près à une série de minis, les fameux PDP-11 de Digital. Lorsque, pour des raisons sans doute voisines de celles déjà évoquées, Digital décida de prolonger sa gamme vers le bas par un micro, baptisé fort joliment le LSI-11, la réalisation des circuits intégrés fut confiée à Western Digital qui produisit ainsi le MCP-1600 : ce micro à quatre puces a été dans un premier temps uniquement réservé à Digital.

En effet, il présente la particularité d'être microprogrammable (voir 3-2) : WD mit donc en ROM le microcode correspondant aux instructions de LSI-11, pour le seul usage de DEC. La microprogrammation permettant une grande souplesse de définition de produit, le 1600 était destiné à toute une série d'avatars (dans le bon et premier sens du terme !).

Le second fut donc baptisé le WD-16. Voisin du LSI-11, mais doté d'instructions supplémentaires, il tenta Alpha Microsystems qui s'en assura l'exclusivité, et en fit la base du micro-ordinateur AM-100 dont on a vu quelques exemplaires à Paris dans des boutiques d'ordinateurs.

Le plus récent fait un pont avec l'actualité : le nom de baptême est maintenant *PASCAL MICRO—ENGINE:* le microcode a donc encore une fois complètement spécialisé le MCP-1600, qui devient l'un des rares concrétisations d'un genre de machine bien souvent décrit sur le papier : une unité arithmétique et logique capable d'exécuter directement un langage évolué. Directement, enfin presque : le Microengine exécute un langage intermédiaire entre le Pascal et l'assembleur, le P-code, sautant donc plusieurs des étages classiques du trajet compilation/exécution que nous rappelons brièvement :

— *schéma normal* : le compilateur traduit le langage évolué en langage d'assemblage, puis l'assembleur traduit ce dernier en langage machine, puis un chargeur charge ce dernier dans l'ordinateur pour exécution.

— *schéma à langage intermédiaire* : le compilateur traduit le langage évolué en un langage de niveau intermédiaire (ici le P-code), puis un traducteur spécial traduit le P-code en langage machine, et il y a enfin le chargeur (cette formule permet une meilleure portabilité du compilateur, dont la structure est rendue indépendante des langages d'assemblage, très différents d'une machine à l'autre : il suffit de réécrire seulement le traducteur pour une machine donnée). Le traducteur peut également être remplacé par un interpréteur qui exécute directement le P-code, mais les expériences effectuées jusqu'ici ont en général donné des vitesses d'exécution très faibles.

Avec le Microengine, l'interpréteur est microprogrammé, ce qui devrait accélérer l'exécution. Comme il ne s'agit cependant que d'un micro, même assez rapide, on peut craindre des temps d'exécution encore longs.

Le plus gros reproche que nous ferions à cette approche est qu'elle est fermée : la machine est spécialisée PASCAL, uniquement. Vous vous en doutez... nous allons être amenés à donner notre opinion sur PASCAL...

C'est beau, c'est structuré, c'est tout simplement de l'Algol revu et corrigé, avec les mêmes défauts. Académique et pur, bien loin des applications pratiques (il faudrait voir un peu les entrées/sorties), c'est parfait pour écrire des systèmes. Le genre BEGIN, BEGIN, BEGIN, BEGIN, chose, END, END, END, END, quoi. Quant aux opérateurs, on est rudement loin derrière APL...
Ce sera tout.

9 - 2. LES NOUVEAUX 8 BITS

Pendant tout ce temps, les micros dits à 8 bits ont pas mal évolué.

La première évolution a été en vitesse. On a par exemple vu apparaître en un premier temps le 8080 A, avec des caractéristiques améliorées, puis des versions tiret 1, 2, etc. avec des fréquences de base passant de 2 à 2,5, puis 3 MHz et plus. De même pour le 6800. Naturellement, les mémoires doivent suivre, et le prix augmente vite.

Puis il y a eu des changements plus sérieux. L'apparition du Z 80 par exemple. Réalisé par des transfuges d'Intel (dont Federico Faggin) qui créèrent Zilog (avec derrière la puissante compagnie pétrolière Exxon), le Z 80 est à première vue attrayant. Compatible avec le 8080 en tant qu'instructions, mais avec des instructions et des registres d'index supplémentaires, il est également plus rapide. Cependant, la plupart des logiciels n'en profitent pas, pour deux raisons essentielles : soit qu'ils aient été écrits auparavant pour le 8080 A, et il est trop coûteux de les modifier pour utiliser les nouvelles instructions du Z 80, soit que les concepteurs veuillent, pour des raisons de sécurité d'approvisionnement, rester compatible 8080 A. Dans ces conditions, la différence de vitesse 8080 A/Z 80 n'est pas très significative.

Par contre, un des avantages du Z 80 a été de donner un signal très utile pour la commande du rafraîchissement des mémoires dynamiques, simplifiant pas mal le travail de conception (si l'on utilise ce type de mémoire). Le Z 80 a également augmenté de vitesse, avec une version à 4 MHz.

Intel a répliqué avec le 8085. Compatible avec le 8080 A, il présente l'avantage d'une vitesse accrue

(5 MHz) et surtout d'être (proportionnellement) moins exigeant en vitesse du cycle mémoire pour une rapidité d'horloge donnée. Le 8080 A pour sa vitesse nominale de 2 MHz exige un accès mémoire de 490 ns maximum, alors que le 8085 atteint 3 MHz en se contentant d'une vitesse mémoire sensiblement identique.

Confronté à cette avalanche de nouvelles possibilités d'un côté et à l'état de tension du marché mondial de l'électronique de l'autre, la réaction du concepteur est tempérée par celle de l'industriel : sécurité d'abord. Conservons un microprocesseur que nous soyons sûrs de nous voir livrer régulièrement, en temps et en heure...

Le 8080 A et le 6800 viennent d'être agréés par le DOD (département de la défense des USA), de même que le 2901 A (tranches bipolaires à 4 bits de AMD, qui ont pratiquement enterré le 3 000 d'Intel en tranches à 2 bits), ce qui va leur assurer à tous une très longue existence (au moins dix ans). Vous ne perdez pas votre temps à bien connaître ces composants !

De nombreux autres 8 bits "classiques" sont également apparus, cantonnés dans des marchés variés souvent définis par des circonstances historiques. Par exemple, le 6502 de MOS Technology, qui a été dès le départ annoncé à un prix très bas, et qui était monté sur une carte didactique très sympathique pour apprendre le 6502 (le KIM-1), a eu pour ces deux raisons beaucoup de succès auprès des amateurs. Il existe dans une version rapide, le 6502 C, qui (à condition que la mémoire soit à 150 ns) exécute un ADD de mémoire à registre en 600 ns, et un JUMP dans 64 K en 900 ns. Sa vitesse est donc de l'ordre de 1,5 millions d'instructions par seconde : pas mal pour un micro 8 bits en MOS !

Enfin, dans la série classique, nous terminerons par le Motorola 6802, directement issu du 6800.

Dans le chapitre 3-6, nous avons donné quelques unes des raisons de notre choix, à l'époque, du 8080 plutôt que du 6800 : les raisons essentielles étaient la disponibilité et certains avantages électroniques. Et pourtant, au départ, le jeu d'instructions de 6800 nous plaisait nettement plus que celui du 8080, surtout sans doute parce qu'il était moins déroutant au premier abord. L'expérimentation (en programmation) venant, le 8080 se révéla faire aussi bien que le 6800 dans l'ensemble des problèmes pratiques posés par la programmation : il était tout simplement handicapé par des mnémoniques d'instructions abominables. En particulier, des instructions au fonctionnement très voisin ont des mnémoniques complètement différents, ce qui occasionna d'ailleurs un certain temps dans la presse américaine des publications de comparaisons 8080/6800 en programmation totalement fausses à l'avantage du 6800, qui paraissait nettement plus rapide, lesdites comparaisons étant redressées quelques mois plus tard par de douloureux rectificatifs... qui amenaient le 6800 et le 8080 ex-aequo en pratique.

Pour en revenir aux aspects électroniques du 6800, deux nous préoccupaient beaucoup : les spécifs extrêmement strictes et complexes de l'horloge, et l'absence de ligne READY. On se rappelle (voir 3-7) que cette ligne permet de mettre le 8080 en état d'attente Tw après T2, si elle est restée basse. Cette caractéristique est très commode, et permet par exemple d'utiliser facilement des RAM lentes et économiques.

Ces points n'ont pas échappé aux concepteurs de Motorola : le 6802 comporte une horloge directement intégrée sur la puce, ce qui est fort sympathique, et une possibilité d'attente évitant d'avoir à changer le rythme de l'horloge (!) pour utiliser des mémoires lentes. Nous en reparlerons avec le 6809.

Tout sur la puce

Parallèlement à cette évolution sont apparus également une série de micros comportant sur leur puce, outre l'unité arithmétique et logique... de la mémoire REPROM ou ROM pour le programme, et un peu de RAM ! Ce sont par exemple chez Intel les 8021/8048/8039 avec 1 ou 2 K de programme, chez Fairchild le 3870 récent (issu du F-8) avec près de 4 K de programme et 128 octets de RAM, et toute une série chez Motorola. La plupart de ces micros sont assez difficiles à programmer, à part le 8048 qui fait exception dans le tas. Citons encore dans cette classe le TMS 1000 à 4 bits, en de nombreuses versions, dont une en C-MOS (par Motorola).

Il devient alors possible d'avoir de minuscules unités centrales complètes, dans un encombrement très réduit, permettant de rendre intelligents les instruments et les engins les plus divers...

Ce mot "difficulté" nous remet en mémoire une petite tirade que nous voulions faire à propos de la documentation fournie par les fabricants de composants pour leurs microprocesseurs. Notons bien que ces remarques n'ont rien à voir avec les qualités des microprocesseurs eux-mêmes...

Dans les premiers temps, la doc Intel nous a paru superbe... parce qu'elle existait ! Force est de dire qu'aujourd'hui, comparée à celle de Motorola, elle est une coudée en-dessous. Disons qu'on trouvera un élément essentiel enterré page 24 dans un petit alinéa... Quant à celle de RCA pour le COSMAC (le 1802 en C-MOS, à consommation ultra-faible, excellent pour le portable) elle est pratiquement illisible, voire à la limite incompréhensible...

Les bizarres

Des micros réalisés pour des marchés très spéciaux commencent à apparaître. Prenons un

exemple : le 2920 de Intel, microprocesseur analogique ! Conçu pour les télécomm, grâce à ses entrées/sorties analogiques avec des convertisseurs numériques incorporés, une RAM de 40 mots de 25 bits, une REPROM de 192 instructions de 24 bits et un cycle de 400 ns, il est capable d'effectuer directement un traitement de signal complexe avec filtration, détection, mixage, sur un signal analogique à 6,5 kHz. Il est en H-MOS, dont nous parlerons plus loin. Plusieurs autres fabricants de composants avancent dans la même voie. Citons par exemple AMI qui a également sorti un microprocesseur de conception voisine, mais travaillant à 17,6 kHz, et Silicon Systems avec son chip récepteur multifréquences.

9 - 3. LES NOUVELLES TECHNOLOGIES : H-MOS, V-MOS ET I3L

Derrière l'irrésistible poussée de l'augmentation constante de densité des circuits intégrés se cache une lutte entre technologies concurrentes, MOS ou bipolaires. Il y en a beaucoup, nous parlerons brièvement de trois d'entre elles, celles qui nous amènent les microprocesseurs à 16 bits de la nouvelle cuvée.

Le H-MOS

C'est l'abréviation de High performance MOS, ce qui se comprend tout seul. Il s'agit là de la technologie la plus récente d'Intel, fondée sur un principe (apparemment) très simple : faire la même chose qu'avant, mais en plus petit... c'est la *contraction de dimensions.*

Le gros avantage est qu'il s'agit d'une évolution directe d'un procédé bien au point, le MOS canal N à grille silicium dont nous avons parlé en détail en 2-4 et 2-5. Le point de départ est la technologie type début 8080, caractérisée par les paramètres suivants :

— longueur du canal : 6 microns (c'est la largeur de la grille) entre source et drain ;
— épaisseur de l'oxyde isolant la grille : 1200 angströms (rappelons que l'angström est le dix millième de micron) ;
— délai de propagation d'une porte : 13 ns ;
— puissance consommée par une porte : 1,5 mW.

A partir de là, on réduit les dimensions : la longueur du canal tombe à 5, puis 4 microns. Bientôt ce sera 3, puis 2 microns, soit des facteurs de réduction linéaire R de 1,5 à 3, entraînant une augmentation de la densité au carré, de 2,25 à 9 ! L'épaisseur de l'oxyde varie comme 1/R, de même que la profondeur de diffusion. La puissance par porte varie en $1/R^2$, comme le délai de propagation, et la tension d'alimentation en 1/R.

Les résultats sont spectaculaires. Un niveau de H-MOS à canal de 3,5 microns demande donc un oxyde de 700 angströms, consomme 1mW par porte, a un délai de propagation de 3 ns seulement, et multiplie la densité par trois ! C'est énorme.

Vous vous en doutez, tout ceci est plus facile à dire qu'à faire, sinon tout le monde y serait...

La première étape est la suivante : jusque vers 4 ou 3 microns, un raffinement continu dans la qualité des masques, de l'alignement et de la photolithogravure sera l'essentiel. Mais des effets fâcheux du second ordre, prenant une importance croissante, commencent à devenir gênants : l'effet de seconde grille et le poinçonnage (voir croquis). Il faut un profil d'impuretés dans le substrat soigneusement choisi par implantation d'ions avec un isolant de grille très mince et très pur. On doit utiliser des dopants à l'arsenic, ce qui entraîne la nécessité d'une énergie élevée pour le faisceau d'ions : la tranche chauffe et risque même de craquer, le refroidissement pose des problèmes. Et ce ne sont pas les seuls... Pas mal de fabricants arrivent à ce stade, tels Mostek avec le procédé Scaled Poly 5 (qui atteint 3 microns), NS avec le X-MOS, Motorola le H-MOS, etc.

La seconde étape, pour descendre à 2 microns, demande en outre un changement plus radical. On atteint en effet une finesse de traits où la diffraction de la lumière gêne la reproduction photographique par contact sur l'émulsion des masques. C'est pourquoi les fabricants les plus avancés s'équipent des nouveaux matériels de lithogravure à projection : l'image du masque est projetée sur l'émulsion de la tranche, le matériel traitant une tranche par minute. Seuls de tels engins permettent d'atteindre les 2 microns en production.

C'est sur un équipement de ce type que compte Intel pour la production de sa nouvelle RAM statique 4 K type 2147 H, dont le temps d'accès garanti est de 35 ou 45 ns selon les modèles ! La fabrication serait en H-MOS II (canaux 2 microns, oxyde de 400 angströms), qui donne donc une vitesse égale ou supérieure à celle du bipolaire. Encore plus rapide, une version évoluée de la 2125, RAM statique 1 K, atteint 15 ns, c'est-à-dire la vitesse de l'ECL ! Le MOS n'a plus de limites, mais au prix d'une complexité de fabrication qui approche maintenant de celle du bipolaire...

Les livraisons de la 2147 H sont promises dernier trimestre 79 : si les délais sont tenus, les concurrents qui n'ont pas encore atteint eux-mêmes la barrière fatidique des 55 ns de la 2147 normale chiffrent l'avance d'Intel non plus en mois, mais en années.

Le V-MOS (et le D-MOS japonais)

C'est la technologie concurrente du H-MOS. Poussée par AMI (American Microsystems Inc.),

et depuis peu par Texas : il s'agit d'une version tri-dimensionnelle du D-MOS.

Ouvrons une parenthèse sur cette dernière technologie : inventée par les japonais (et apparemment dédaignée par les américains), elle permet d'obtenir les avantages d'un canal court... sans avoir à le fabriquer !

Le principe du procédé est le suivant : par une première diffusion, on réalise des transistors à appauvrissement (voir 2-5) avec une taille de canal confortable, 5 microns par exemple. Puis par une seconde diffusion (de bore) on intègre un canal à enrichessement large de 1 micron, par auto-alignement (voir grande planche 2-5 pour le principe de l'auto-alignement) sur l'ouverture source-jonction. D'où le nom du procédé, DSA-MOS (diffusion-self-aligned-MOS), abrégé depuis en D-MOS. En somme, n'étant pas capable de tracer des lignes assez fines, on contourne la difficulté.

Le V-MOS est une amélioration sur ce thème : abandonnant les procédés plans, on creuse par attaque chimique des sillons à section triangulaire avec des flancs à forte pente (plus de 45°) à travers les couches épitaxiales (de haut en bas : la couche π de 1 micron, légèrement dopée P, la couche p d'un peu moins de 1 micron, plus fortement dopée, et le substrat de type N).

L'arête du dièdre pénètre juste dans le substrat, qui joue le rôle de la source. Le drain est diffusé en surface, et le canal est donc sur la pente du sillon. On gagne ainsi doublement de la place : la source est sous le drain, et non à côté, et la forte pente diminue la taille du canal (en projection à plat). On gagne 40 % par rapport aux procédés classiques.

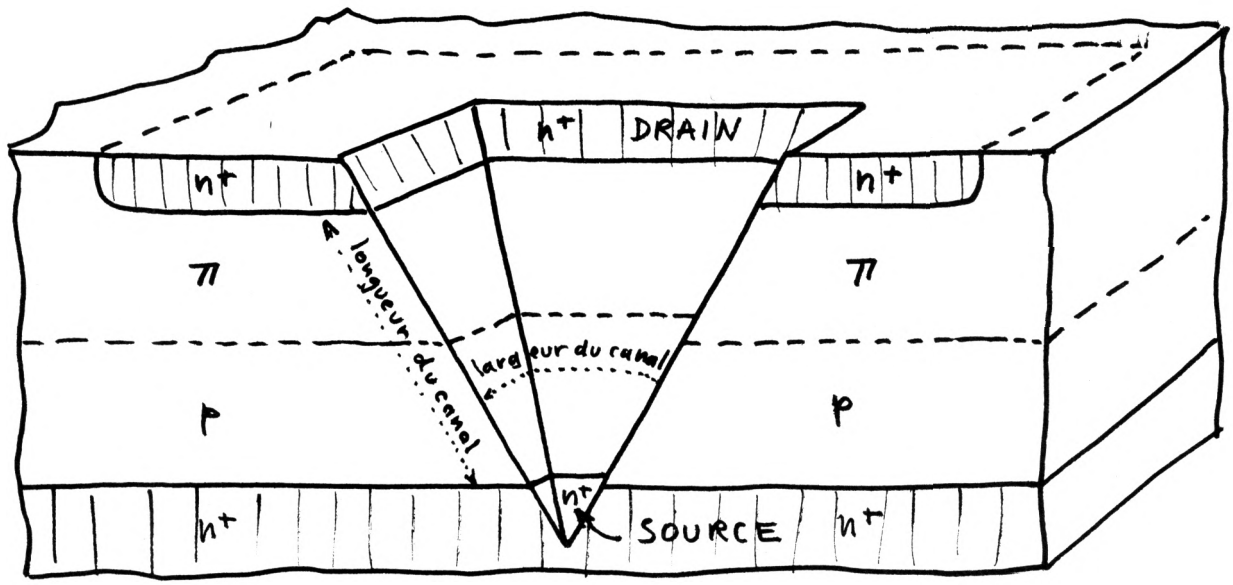
Les avantages et les inconvénients sont les mêmes que lorsque vous achetez une parcelle de terrain bien en pente sur la côte d'azur : vous avez en réalité davantage de mètres carrés que sur le plan cadastral, mais la construction est difficile et coûte rudement cher !

Les cellules de mémoire statique sont très jolies : leurs transistors sont de petites pyramides la pointe en bas, avec une base (de la pyramide !) de 15 microns. Une puce de 1 K, avec la cellule classique à 6 transistors, comprend environ 7.500 composants, et la plus petite taille réalisée ne fait que 2,8 mm², ce qui donne 2.600 composants au millimètre carré. Il semble que la production pourrait se faire sur une puce de 3,6 mm² contre 6 mm² (disent les partisans du V-MOS) pour un procédé H-MOS.

L'avenir seul nous apprendra la vérité, et l'heure de vérité sonnera sans doute courant 1979. En effet, Texas Instruments (qui a un retard manifeste en RAM ultra-rapides) poursuit activement depuis deux ans deux programmes de rattrapage de développement en parallèle : H-MOS et V-MOS. Comme même TI ne peut se permettre de dupliquer longtemps un tel effort, une décision lourde de conséquences pour l'avenir du V-MOS sera probablement prise fin 79 : pour l'instant, les deux technologies sont si proches en résultats que l'issue est encore incertaine, ce qui doit bien embarrasser le management.

L'I3L et le Fairchild 9440

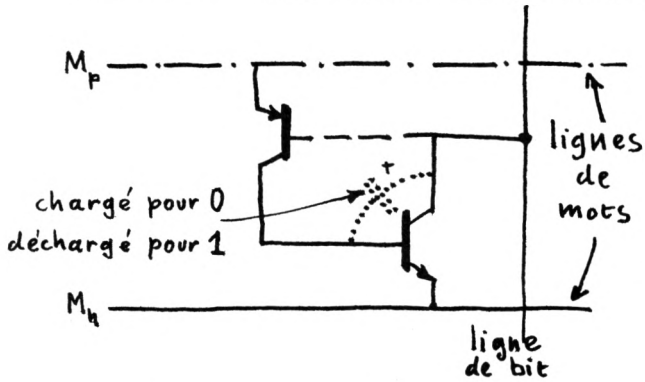
Brièvement mentionnée en 2-6, l'I2L, logique à injection intégrée, est une technologie bipolaire,

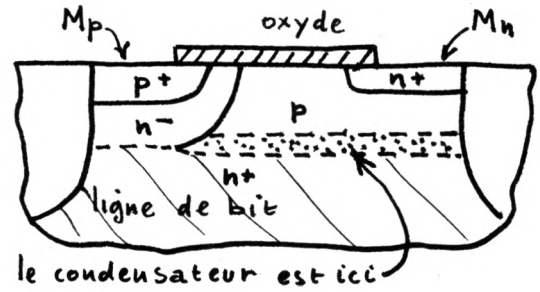


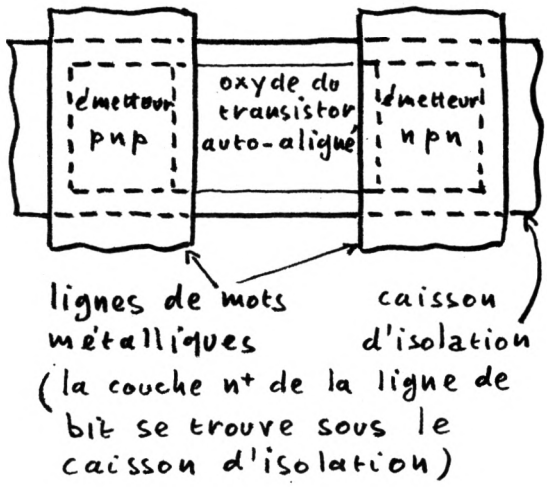
Structure V-MOS

donc de la grande famille TTL. Son dernier développement chez Fairchild est l'I3L, c'est-à-dire l'I2L à isolement isoplanaire. Un circuit logique en I3L a une surface de 600 à 2000 microns carrés, un délai de propagation inférieur à 5 ns... et une consommation pouvant descendre à 1 nanowatt ! Très indiqué pour le militaire et le spatial, et de plus excellente résistance aux radiations ionisantes.

Cellule mémoire I³L dynamique bipolaire. Le transistor pnp/npn est vu en coupe et en plan. La charge est stockée dans la jonction partagée collecteur-base de ce double transistor.







L'I3L a permis en particulier la fabrication de la première RAM dynamique bipolaire, la 93481 de Fairchild. La version A a un accès de l'ordre de 100 ns, une capacité de 4 K, une consommation de 70 mW en attente et 450 mW en activité. La puce fait 2,5 sur 3 mm. Des puces plus récentes à 16 K descendent à 400 microns carrés par cellule.

L'un des avantages de l'I3L est de permettre de faire cohabiter sur un même substrat des structures complémentaires npn et pnp, plus ou moins entremêlées, permettant des combinaisons intéressantes, telles que des transistors pnp à double diffusion couplés à un npn et jouant le rôle de cellule de mémoire dynamique de petite taille.

Le Fairchild 9440, joliment appelé Microflame, est un des rares micros fabriqués en I3L. Issu de la ligne NOVA, qui décidément a eu beaucoup de succès, il est donc capable d'exécuter les instructions NOVA. C'est le premier des 16 bits de récente génération, et il est un tout petit peu plus rapide que le MicroNOVA. Sa conception est assez particulière. Une seule tension d'alimentation (+ 5 V) et un interfaçage plus facile que son concurrent. Mais le bus interne de données sur la puce est large de 4 bits seulement, ce qui force à 4 transferts par mot, d'où la vitesse un peu faible. Fairchild prépare une deuxième version à bus interne de 16 bits cette fois, qui sera environ deux fois plus rapide : ce sera le 9445. Performances annoncées : avec une horloge à 20 MHz, multiplication de 16 bits par 16 bits en 5 microsecondes.

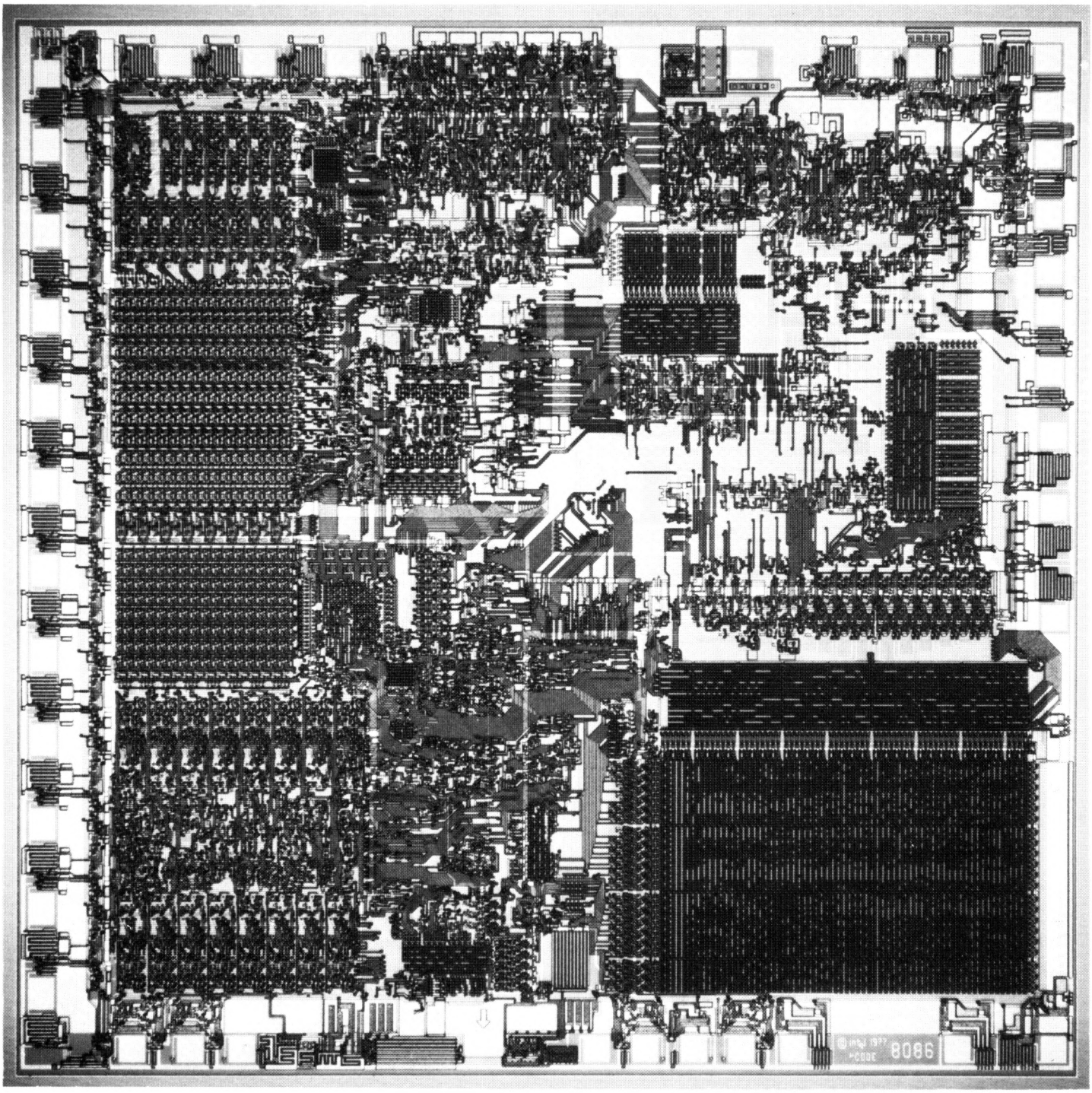
9 - 4. LE PREMIER DES NOUVEAUX 16 BITS : LE 8086

Nous rentrons maintenant dans le domaine des micros de grande cuvée. Le produit 16 bits d'Intel était très attendu : il a été mis sur le marché à la mi-78, la production en volume intervenant en 79.

9 - 4 - 1. Technologie du 8086

C'est le premier microprocesseur fabriqué en H-MOS, à appauvrissement, la dimension de canal adoptée étant de 4 microns. Cette technologie, déjà utilisée pour la 2147, RAM statique de 4 K très rapide (55 ns), a permis pour le 8086 de loger 29000 transistors sur une puce carrée d'assez grande dimension, puisqu'elle faisait 5,7 mm de côté au départ de la production, ce qui donne environ 1000 composants par mm² (attention, ne pas comparer avec les densités obtenues pour les mémoires qui sont nettement plus élevées, vu la régularité de la structure). La boîtier est un Cerdip à 40 pattes.

A côté des 6000 transistors du 8080, nous avons un processeur tout à fait impressionnant. De plus, le H-MOS a permis d'arriver à des délais de propa-



Les 29 000 transistors du 8086. Le copyright en bas à droite est de 1977. Le grossissement linéaire est de 32 fois. La complexité est cinq fois plus grande que celle du 8080. Le MC 68000 de Motorola aura, lui, encore deux fois plus de composants que le 8086...

gation de 2 ns par porte sur la puce, aussi courts que ceux obtenus en technologie TTL Schottky, beaucoup plus coûteuse à fabriquer. La fréquence d'horloge s'établit donc à 5 MHz pour le 8086 normal (200 ns) et 8 MHz pour les puces sélectionnées rapides (125 ns). Dès sa sortie, le 8086 s'est donc placé comme le plus rapide des 16 bits monopuce disponibles. Les vitesses de mémoire nécessaires sont respectivement pour les deux fréquences de base de 800 et 500 ns pour le cycle, et de 460 et 295 ns pour l'accès, ce qui reste raisonnable et dans une gamme de prix abordable : Mostek, Siemens et NEC le produiront en seconde source dès que possible.

Le 8086 est conçu comme une extension logique du 8080 : il s'agissait d'une quasi-obligation pour Intel, vu l'importance de la clientèle 8080. Donc la ligne directrice est : donner mieux, davantage, dans le même style. Nous allons voir comment.

9 - 4 - 2. Extension de l'architecture

Elle est toujours à registres généraux, comme le 360, le 370 ... et le 8080. Il y a cette fois-ci deux séries de registres de 16 bits :

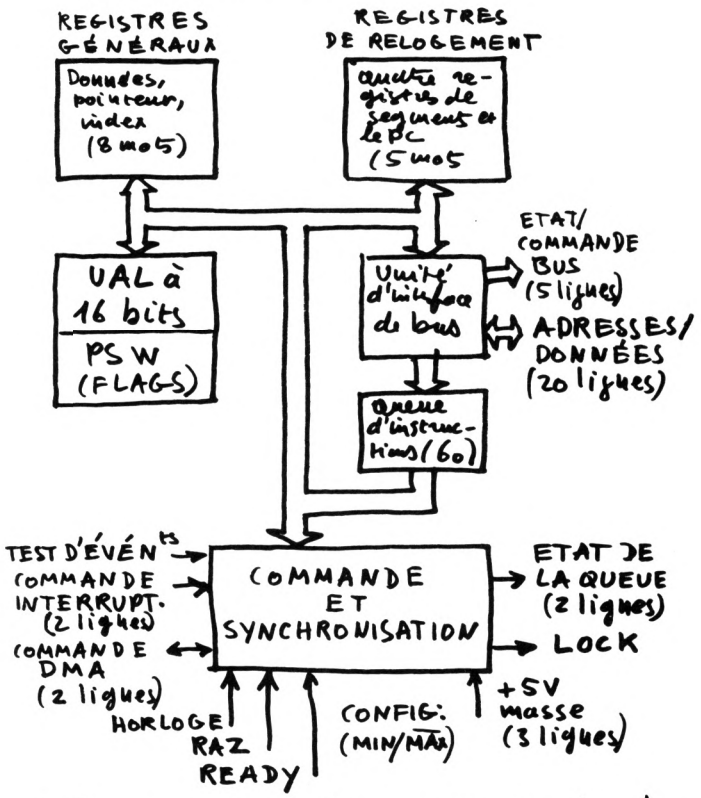


Schéma général du 8086 : les broches adresses peuvent fournir d'autres informations par multiplexage ; par strappage on peut changer certains signaux.

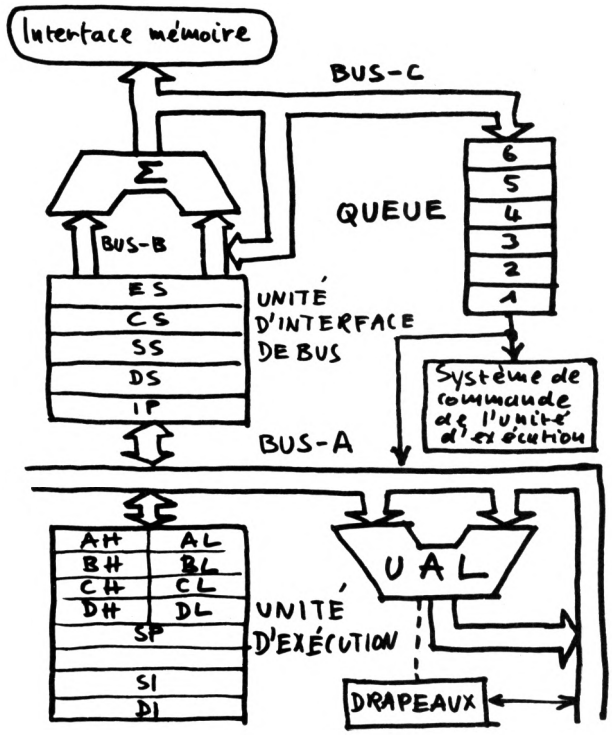
— la série des registres généraux : 4 registres (également adressables par moitié) pour le déroulement de l'arithmétique et des instructions logiques, 2 registres d'adresse servant de pointeurs pour la mémoire (base et pile), 2 registres d'index, le compteur ordinal et un registre de drapeaux d'état. Au total, près du double du 8080, donc une puissance très accrue, et surtout enfin de vrais registres d'index,
— la série des registres de segments : étendant considérablement les possibilités d'adressage, ils permettent d'adresser 1 million d'octets de mémoire centrale (au lieu de 64 K).
Ils peuvent stocker les adresses de relogement de 4 segments de 64 K simultanément (programmes ou données).

L'extension de l'adressage est une caractéristique qui nous paraît plus importante que l'augmentation de la puissance de traitement : c'est elle qui ouvre maintenant au micro le domaine du mini, juste au moment où le coût de la mémoire commence à baisser sérieusement, permettant de passer facilement à des capacités importantes.

9 - 4 - 3. La queue

Il s'agit là d'une des caractéristiques les plus intéressantes du 8086 : le fameux système d'*antémémoire*, ou *look-ahead*, l'œil de lynx qui scrute au loin le programme, jadis réservé aux ordinateurs les plus puissants du monde ! *Sic transit...*

Le processeur comprend deux unités : l'unité d'exécution EU (qui comporte elle-même 7 registres de la première série et l'unité arithmétique



et logique), et l'unité interface du bus BIU (qui comprend le compteur ordinal IP, les 4 registres de segments et la queue d'instructions).

Cette queue fait 6 octets de long. Chaque fois que la mémoire est inactive et qu'il y a au moins 2 octets de libre dans la queue, le BIU va chercher à l'avance les instructions suivantes du programme, en séquence. Dès que l'exécution de l'instruction en cours est terminée, l'EU s'approvisionne de la première instruction de la queue, à grande vitesse, sans être ralentie par un accès mémoire. La vitesse de l'EU est assez grande pour que par exemple deux instructions d'un octet puissent s'exécuter en recouvrement complet pendant un cycle d'accès mémoire.

La longueur optimale à donner à cette queue a été déterminée par simulation de programmes réels, d'où le chiffre de 6 octets. La taille des instructions étant de 2 octets en moyenne, la longueur de queue est suffisante pour garder l'EU occupée à peu près en permanence. Naturellement, quand on tombe sur un branchement, le reste du contenu de la queue est perdu, et le BIU doit la réalimenter : une queue trop longue amènerait ainsi à un gaspillage d'accès mémoire.

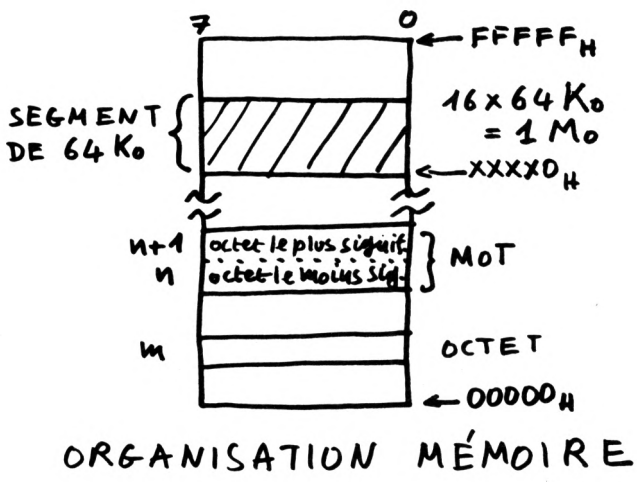
Ce dispositif accélère donc beaucoup l'exécution des programmes, augmentant considérablement la puissance de traitement du 8086. Accessoirement, remarquons qu'il rend difficile le calcul du temps exact de déroulement d'un programme, le temps d'exécution d'une instruction dépendant de sa présence ou non dans la queue à ce moment. Les temps publiés par Intel supposent que l'instruction est dans la queue, mais c'est son temps d'exécution le plus long qui est pris : on a ainsi un cas moyen. Le temps le plus long possible en pratique est obtenu en faisant la somme de ces temps et en ajoutant 10 % de marge pour tenir compte des cas de queue vide. Dans la pratique, s'il y a un temps de boucle critique à respecter, il vaudra mieux la dérouler 5 ou 10 millions de fois et la chronométrer. C'est un inconvénient mineur par rapport au gain de performances.

9 - 4 - 4. L'adressage

L'espace mémoire adressable est de un million d'octets (plus précisément 1048576 octets, ou 1024 K), ce qui représente une adresse de 20 bits, allant de zéro à 1048575, ou FFFFF en hexa.

Le processeur n'ayant que des registres de 16 bits, le truc pour se promener dans notre méga octet est le suivant : nous pouvons adresser directement dans des segments de 64 K, dont l'origine est obligatoirement à une adresse multiple de 16 (se terminant donc par 0000 en binaire, ou par 0 en hexa).

A un instant donné, nous pouvons sans manipulation particulière travailler dans 4 segments courants, dont les adresses de départ auront été



ORGANISATION MÉMOIRE

chargées dans les 4 registres de segments (il s'agit bien sûr de 16 bits les plus significatifs desdites adresses, les 4 de droite étant toujours zéro). Ces 4 segments portent les noms suivants :

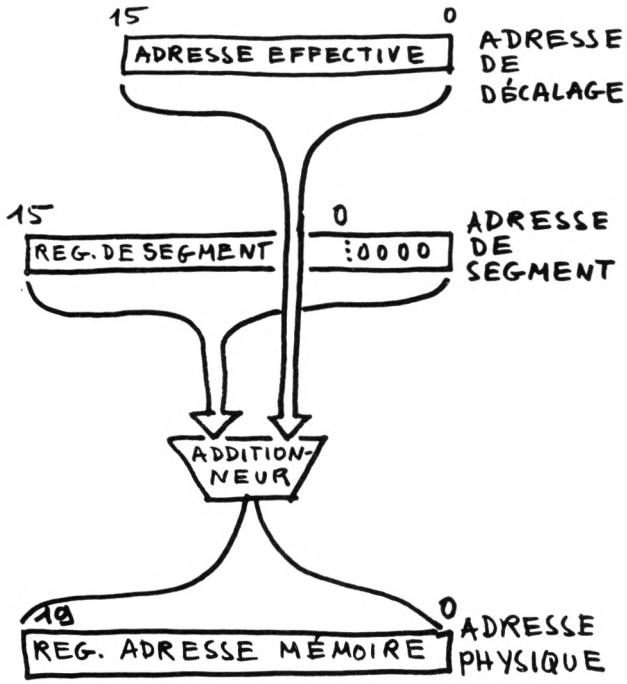
— segment programme courant (CS, code),
— segment données courant (DS, data),
— segment pile courant (SS, stack),
— segment supplémentaire courant (ES, extra).

Remarquons bien que les restrictions d'emplacement indiquées n'imposent pas que les segments soient contigus : ils peuvent se recouvrir partiellement, être disjoints, etc. Par ailleurs, en chargeant d'autres adresses dans les registres de segments on peut adresser d'autres segments. S'ils sont adjacents, 16 segments couvrent les 1024 K.

Si nous nous abstenons de telles manœuvres (chargement ou manipulation de registres de segments) dans un programme, il sera dynamiquement relogeable, c'est-à-dire que nous pourrons interrompre son déroulement, le déplacer en mémoire pour l'amener à un nouvel emplacement, puis, après avoir chargé les nouvelles adresses dans les registres de segments, relancer son exécution sans lui avoir changé quoi que ce soit.

En effet, lorsqu'un programme se déroule, les adresses générées par indexation, adressage indirect, immédiat ou n'importe quel autre mode ont toujours 16 bits de long, et se déplacent donc dans un espace de 64 K en adressant par rapport à son origine. Lorsque le contenu du registre de segments leur est automatiquement ajouté, juste avant l'exécution, décalé de 4 bits à gauche, donnant une adresse physique de 20 bits dans le registre, on obtient bien automatiquement une translation de *toutes ces adresses*. C'est même le cas si l'adresse est obtenue par calcul et modification directe du programme par lui-même (pratique peu recommandable par ailleurs...).

Cette translation fait intervenir des registres de segments différents selon le cas. S'il s'agit d'une adresse d'instruction (cas d'un branchement par



exemple), c'est le contenu de CS qui s'ajoute. S'il s'agit d'une adresse de donnée, c'est le contenu de DS. Pour les opérations de mise sur pile et de désempilage, par des CALL par exemple, c'est le contenu de SS. Dans le cas d'instructions sur chaînes de caractères faisant intervenir le registre index destination (DI), c'est le contenu de ES qui s'ajoute aux adresses de chaînes.

Ce sont là les principales interventions par défaut (automatiques, donc) des registres de segments. En utilisant un préfixe de forçage d'un octet placé devant une instruction, le programmeur peut changer ces utilisations automatiques de registres en désignant ainsi lui-même (à ses risques et périls !) le registre de segment à utiliser.

Une programmation évoluée, puissante et complexe. Attendons la suite.

9 - 4 - 5. Le jeu d'instructions et les modes d'adressage

Nous n'entrerons pas dans le détail de la façon dont les instructions manipulent les registres généraux ou de drapeaux, l'accumulateur, les pointeurs, etc. Bornons-nous à remarquer quelques points intéressants.

Tout d'abord, les 5 modes fondamentaux d'adressage se développant en 24 variantes... Dans le cas usuel, on adresse les opérandes soit directement avec l'adresse normale dite *adresse de décalage* de 16 bits (elle décale par rapport à l'adresse origine de segment), soit en adressage immédiat, soit indirectement à l'aide des registres de base BX ou BP, ou des registres d'index SI ou DI, ou d'une combinaison d'un registre de base et d'un registre d'index qui donne un déplacement de 8 ou 16 bits selon le cas de figure.

On a donc un fonctionnement d'instruction généralisé à deux opérandes : l'un des deux sera un élément en mémoire ou un registre quelconque, et l'autre sera un registre ou une constante (opérande immédiat) dans une instruction. Le résultat produit par l'instruction sera remis dans l'emplacement d'un des opérandes de départ (à l'exception de l'opérande immédiat). Naturellement, il y a aussi un bon choix d'instructions à un seul opérande, avec la même possibilité de remise du résultat dans l'emplacement de départ (sauf opérande immédiat également).

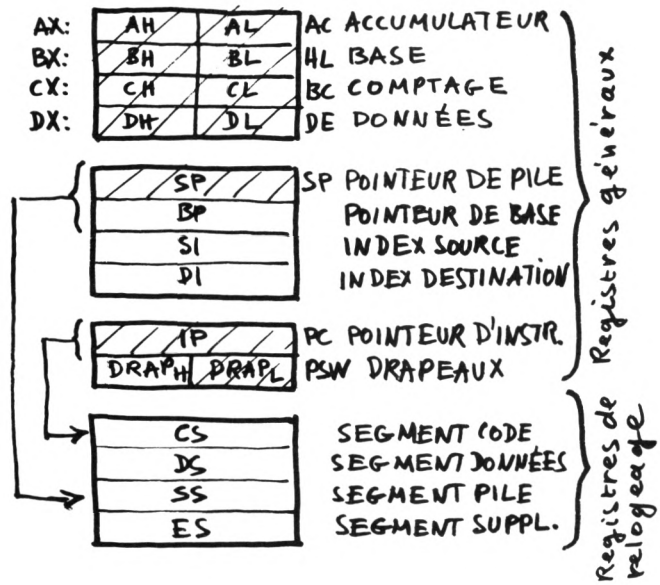
Pratiquement toutes les instructions sont en deux variantes : opérandes 8 ou 16 bits.

L'arithmétique est beaucoup plus développée que sur le 8080, car en effet les 20 instructions correspondantes permettent le travail :

— sur 8 ou 16 bits, donc,
— en signé (complément à 2) ou non signé,
— complet (les 4 opérations),
— en décimal sur du condensé ou de l'étendu, par correction.

De façon générale les objets traités ont 8 ou 16 bits, et un nombre limité d'instructions traite des pointeurs de 32 bits (composés d'une adresse de décalage et d'une adresse de segment, soit 16 plus 16). Il y a des instructions de transfert de blocs.

Deux sous-groupes d'instructions de branchement (sauts, appels et retours) permettent l'un d'opérer à l'intérieur d'un segment, l'autre d'aller du segment courant à un autre segment qui dès cet instant deviendra le nouveau segment cou-



Le sous-ensemble hachuré est celui du 8080

rant. Les 16 types de branchements utilisent la variété de modes d'adressage déjà vus.

Nous conclurons sur un type particulier d'instructions : destinées à travailler sur des chaînes d'octets ou de *mots* de 16 bits, il est possible de les faire exécuter soit normalement, soit à répétition par un préfixe spécial déclenchant une itération. Si ces instructions peuvent être aussi courtes, c'est bien sûr qu'elles font implicitement intervenir des registres déterminés, y compris l'accumulateur.

Voici un exemple de la façon dont ces instructions opèrent, sur un cas particulièrement utile en cette époque où les micro-ordinateurs se mêlent des affaires de leurs grands frères en empiétant quelque peu sur leur terrain : la traduction de chaînes de caractères du code EBCDIC (code IBM issu du monde de la carte perforée) en code ASCII universel (issu du monde des télécomm et des minis/micros).

Le programme de traduction s'écrit en assembleur de la façon suivante :

```
            JCXZ        SUITE
BOUCLE      LODB
            XLAT
            CMP         AL,04H
            STOB
            LOOPNE      BOUCLE
SUITE       ..........  ..........
```

Ce programme est assurément très court, mais plus encore que vous ne le croyez : la boucle ne fait que 7 octets de long ! Voici tous les détails du mystère.

La chaîne de caractères en code EBCDIC à traduire se trouve dans le tampon en mémoire d'adresse symbolique TAMEBC. Elle se termine par le caractère EOT (fin de texte).

On charge TAMEBC dans le registre index source SI.

La table de traduction EBCDIC/ASCII est à l'adresse TABLE. On charge TABLE dans le registre d'adresse de base BX.

La chaîne de caractères traduite en SCII sera mise en mémoire en TAMASC. On charge TAMASC dans le registre index destination DI.

On initialise le registre de comptage CX en le chargeant de la longueur effective de la chaîne à traduire. Ça y est, tout est prêt.

Première instruction (de deux octets de long) : elle teste le compteur de CX. S'il est nul, il n'y a pas lieu de traduire, et on saute à la suite, à l'adresse SUITE.

Deuxième instruction : c'est l'une de ces instructions compactes de chaîne, elle ne fait qu'un octet de long. Elle charge dans AL l'octet sur lequel pointe SI (une variante permet de manipuler mot par mot), puis SI est automatiquement incrémenté de un (de deux si c'est la variante mot).

Troisième instruction : idem, elle ajoute au contenu de BX la valeur contenue en AL, ce qui donne une adresse pointant sur un octet contenu dans la table de traduction (de 256 octets de long). L'octet en question est chargé dans AL, écrasant le contenu précédent : c'est notre configuration ASCII.

Quatrième instruction (de deux octets de long) : elle compare le contenu de AL avec l'opérande immédiat 04, exprimé en hexadécimal (c'est la valeur du caractère fin de texte EOT en ASCII, soit 00000100). S'il y a égalité, le drapeau d'égalité est positionné.

Cinquième instruction : à nouveau d'un octet, elle stocke en mémoire le contenu de AL à l'adresse où pointe DI, et progresse DI.

Sixième dernière : de deux octets de long, intitulée "boucler tant que pas égal", elle teste le drapeau d'égalité et, s'il n'est pas positionné, branche 6 octets plus haut à LODB. Si par contre le drapeau est positionné, on va en séquence à SUITE.

Cet exemple montre bien à quel point les instructions de manipulation de chaîne de caractères sont bien pensées. La commande de la progression est faite ici par boucle en logiciel, on peut aussi la faire par matériel avec les préfixes. Tout ceci permet de réaliser des instructions BASIC de traitement de chaînes très efficaces pour la gestion.

9 - 4 - 6. Interruptions, multitraitement

Nous ne rentrerons pas en détail dans la description des mécanismes correspondants, et nous nous bornerons simplement à quelques remarques. Le système classique du 8080 que nous avons étudié est ici étendu à 256 niveaux directement vectorisés (voir 3-7 instructions relatives aux interruptions). Il est également possible de détecter commodément des conditions anormales en exécution de programme (comme un débordement imprévu par exemple, ou une erreur dans une division) : une instruction d'un octet provoque une interruption si la condition apparaît.

Des mécanismes particuliers sont prévus sur le 8086 pour permettre des systèmes multiprocesseurs : un préfixe spécial d'un octet peut être placé en tête de n'importe quelle instruction. Dans ce cas, le processeur positionne une ligne externe de verrouillage pendant toute la durée de l'instruction, la ligne LOCK (active basse). Ceci permet, grâce à des circuits internes convenables, d'interdire l'accès au bus pendant toute la durée correspondante : par exemple, pendant l'exécution XCHG (échanger contenu registre et mémoire) qui permettra de positionner un sémaphore de blocage en mémoire tout en amenant sa valeur précédente pour la tester dans l'accu. Si en effet

un autre processeur par le plus malheureux des hasards procédait à une prise de contrôle du bus justement entre deux cycles de XCHG, pour lui-même manipuler le sémaphore, on tomberait dans la fameuse étreinte mortelle chère à Edsger W. Dijkstra où les processeurs, tels deux chiens de faïence électroniques, gronderaient sur place à cinq mégahertz, leurs yeux rouges fixes, jusqu'à ce que les centrales nucléaires fournissant l'électricité tombent en poussière après la consommation des siècles...

9 - 4 - 7. Le pont micro-mini

Finalement, à la lumière caractéristique que nous venons de voir, le 8086, de même que les autres 16 bits à venir, nous promet une nette progression sur le pont qui va du micro au mini (tout en conservant dans son cas particulier une compatibilité poussée avec le 8080 au niveau du source, chaque instruction de 8080 ayant un équivalent 8086).

La technologie (qui entraîne l'augmentation de fréquence d'horloge), l'architecture étendue (avec la queue d'instructions), l'arithmétique plus étoffée, l'indexation et les nouveaux modes d'adressage, tout concourt à beaucoup accélérer la vitesse. Selon les types de programmes, le 8086 est de 7 à 12 fois plus rapide que le 8080 A, et les programmes de 10 à 25 % moins encombrants. C'est là un progrès considérable. En reprenant les comparaisons avec des ordinateurs traditionnels (3-7 Petite programmation pondérée), on se rend mieux compte du progrès : on est maintenant à mi-chemin entre le 360/40 et le 360/50 !!

Avec un bon logiciel par là-dessus, *quo non ascendet ?*

9 - 5. LES 16 BITS QUI ARRIVENT

Ils sont nombreux, nous retiendrons les deux plus importants pour les décrire dans les grandes lignes.

9 - 5 - 1. Le Z-8000

L'équipe de Zilog a innové, rompant résolument avec le passé. Et tout d'abord, le sigle Z-8000 cache deux microprocesseurs, dont l'un est un sous-ensemble de l'autre.

La première version, en boîtier à 48 broches... a 23 lignes adresse (24 selon d'autres documentations), et peut donc adresser 8 (ou 16) millions d'octets de mémoire centrale ! Comme de plus programme système et programme utilisateur ont chacun des zones code, données et pile distinctes, ces capacités seraient multipliées par six, atteignant 96 millions d'octets de mémoire centrale ! Qui dit mieux... Les premiers échantillons du Z-8000 ont été livrés en mars 79. Leur puce comprend 17 500 transistors, et les secondes sources seront AMD et SGS-Ates.

Là aussi nous retrouvons le schéma des segments de 64 K (appelés ici sections), gérés par un boîtier externe, le MMU (memory management unit) qui, de façon transparente, s'occupe de la segmentation et de la protection mémoire. La doc parle d'adresses virtuelles (qui n'en sont pas) au lieu d'adresses de déplacement, et avec les "sections" en plus, on tend à partir en brasse coulée.

L'une des origines de la confusion semble être que les numéros de sections étant en 7 bits, il y a 128 sections, et comme elles font 64 Ko chacune, le total est de 8 Mo, alors que l'adressage physique sur 24 bits va de 0 à FFFFFFH, c'est-à-dire 16 Mo, les sections ne pouvant tout couvrir.

Le processeur comprend 16 registres généraux de 16 bits, subdivisés ou groupés logiquement en entier ou par moitié en :
— 16 registres d'un octet, ou
— 16 registres de 16 bits, bien entendu, ou
— 8 registres de 32 bits, ou
— 4 registres de 64 bits.

Tous les registres à 8 ou 16 bits sont utilisables comme accumulateurs, et 15 des 16 registres de 16 bits peuvent être utilisés comme registres d'index. Un certain nombre d'entre eux sont cependant réservés comme pointeurs de piles, pointeus de début de sections, etc. Un registre d'état de 64 bits contient le compteur ordinal (16 + 7 bits) et un paquet de drapeaux. Enfin, un compteur pour le rafraîchissement des mémoires dynamiques complète de tableau.

La fréquence de base de l'horloge qui anime tout l'ensemble est de 4 MHz.

Le Z-8000 peut travailler sur 7 types de données :
— des bits,
— des chiffres DCB (4 bits),
— des octets,
— des mots (16 bits),
— des mots longs (32 bits),
— des chaînes d'octets (8n + 8 bits),
— des chaînes de mots (16n + 16 bits).

En réalité, l'accès vrai ne descend pas au-dessous de l'octet, et on fait appel à des instructions spéciales pour manipuler des éléments plus petits, telles SET B (positionner un bit) ou DAB (ajustage décimal, comme le DAA chez Intel).

Il y a 8 modes d'adressage :
— registre : la valeur de l'opérande est le contenu du registre,
— registre indirect : elle est en mémoire à l'adresse qui est dans le registre,
— adresse directe : idem, adresse dans l'instruction (mode normal),
— immédiat : l'opérande est dans l'instruction,
— indexé : l'adresse dans le registre décale celle de l'instruction,
— adresse de base : l'adresse dans l'instruction décale celle du registre,
— index de base : l'adresse est obtenue par addition de deux registres,

— adresse relative : relative à la valeur courante du compteur ordinal,

ce qui au premier examen paraît être du même tonneau que le 8086, en tout cas aussi complet... et compliqué ! On retrouve un de ces parfums de 360/370... les formats RX, RR, tout y est, il ne manque que le SS : ce sera manifestement pour la prochaine génération, et c'est dommage, car c'est le plus commode.

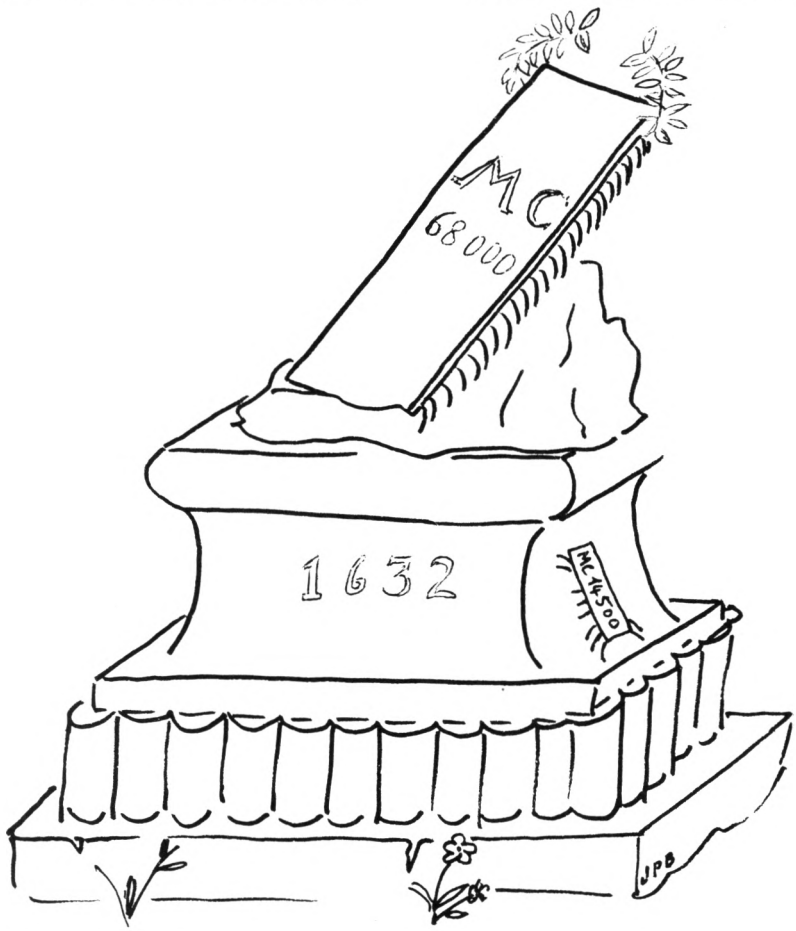
La vitesse, maintenant. Là, la comparaison avec le 8086 va être bien difficile, à cause de son look-ahead qui n'existe pas sur le Z-8000. En tant que déroulement d'instructions isolées simples, l'augmentation de vitesse du Z-8000 standard par rapport au 8080 A standard (horloge à deux mégas) est mesurée par un facteur qui va de 3,3 à 3,5 fois pour le chargement d'un octet ou d'un mot dans un registre, et atteint 6,6 fois pour une addition sur un mot (16 bits). On ne peut cependant pas en conclure grand-chose, et il faudra attendre des essais pratiques pour avoir des idées plus précises sur ce point important. Il est malgré tout déjà apparent que le Z-8000 a une rapidité du même ordre que celle d'un PDP 11/45, ce qui n'est pas une mince performance !

Il y a une autre différence importante avec le 8086. Le Z-8000 a un jeu d'instructions privilégiées (entrées/sorties, manipulation de registres de commande) qui ne peuvent être exécutées que par un système d'exploitation, donc dans l'état système du processeur. Comme vu (3-7, De la simplicité), ceci nous lance tout droit dans le domaine logiciel des minis et des lourds, avec l'operating system gérant les ressources et les entrées/sorties pour tous les programmes s'exécutant en multiprogrammation, résolvant les conflits, détectant les accès hors des partitions, etc. Bref, tout le tremblement...

9 - 5 - 2. Le Motorola MC 68000 : vers les 32 bits

Là, c'est la bête, dont les premiers échantillons sont arrivés mi-79. Seize registres de 32 bits ! Un adressage direct sans besoin de segmentation jusqu'à 4 Mo ! Et 16 Mo avec le boîtier de gestion mémoire... C'est vraiment formidable, et un rêve pour la programmation. Nous attendons le 68000 avec le plus grand intérêt, vu également les accords de fabrication passés en France avec Thomson-Efcis.

Ce micro, en fait, est plus qu'un 16 bits normal : c'est un intermédiaire vers une vraie architecture à 32 bits, du style gros mini, qui pourrait paraître vers 1982. La puce contient cependant déjà le nombre colossal de 68000 transistors ! Lui aussi



possède un jeu d'instructions privilégiées, destinées à faciliter la multiprogrammation au travers d'un système d'exploitation.

9 - 6. LES UN BIT !

Mais oui, ça existe ! Tout à l'autre bout de l'échelle, nous trouvons le MC 14500 B de Motorola qui, décidément, couvre la gamme la plus vaste possible.

On sent dans ce genre de produit une réminiscence des puces de calculatrices de poche, ainsi que du un bit Burroughs, appartenant à la famille des microprocesseurs booléens.

Le MC 14500 B se présente en boîtier à 16 pattes : il est en technologie C-MOS, donc à très faible consommation, ce qui est intéressant pour des produits portables. La bonne résistance au bruit du C-MOS peut également être un argument valable pour certaines applications.

Ce micro est le plus simple imaginable : il comporte essentiellement l'horloge sur la puce, un registre instruction et un accu à un bit qui est une simple bascule D. Les instructions font 4 bits ! Il y a donc 16 instructions en tout, y compris le NOP. Elles sont du style charger, charger en complémentant, OU, NON OU, NON OUX, saut de la prochaine instruction, addition avec sortie de retenue, etc.

On a donc affaire à une machine, fonctionnant à 1 MHz, permettant par exemple de traiter une chaîne de bits extraite au fur et à mesure d'un registre à décalage, et de remettre les résultats dans un autre registre à décalage. Pour mettre les programmes au point, on peut utiliser un autre micro : par exemple, un 6800 prendra dans sa mémoire centrale une série d'instructions de 4 bits et les enverra au 14500 via une patte de son PIA (boîtier similaire au 8255).

Un tel microprocesseur est très près dans son esprit de la logique câblée, et à ce titre pourra, moins que d'autres, dépayser les électroniciens "classiques". On peut par exemple réaliser un contrôleur de feux tricolores de circulation avec 10 petits boîtiers au total, y compris les timers.

Cependant, la baisse continue du prix des 8 bits, l'apparition de RAM organisées sur 8 bits (permettant de diminuer le nombre de boîtiers), ainsi que celle des microprocesseurs comprenant RAM et ROM ou RPROM sur la même puce, va sans doute restreindre progressivement le domaine valable des microprocesseurs à un bit.

9 - 7. LES PLUS QUE 8 BITS

Une question d'actualité : les 8 bits sont-ils morts ? Ce n'est en tout cas pas l'opinion des ingénieurs de Motorola qui ont conçu un 8 bit de structure très avancée, le 6809, ni de ceux d'Intel qui viennent d'annoncer, suivant d'ailleurs une démarche diamétralement opposée, le 8088 dont nous parlerons tout à l'heure...

9 - 7 - 1. Le 6809

Il s'agit là d'une évolution, sans doute ultime, du fameux 6800, l'un des deux standards de l'industrie en micros 8 bits. Nous avons déjà cité le 6802, mais le 6809, lui, va beaucoup plus loin. Les deux buts principaux de conception ont été : mettre sur le marché le meilleur 8 bit possible, tout en gardant la compatibilité avec le 6800.

Cette dernière condition aurait été beaucoup trop restrictive si elle avait été exigée au niveau du langage machine lui-même : fort heureusement, une enquête auprès de la clientèle permit de se rendre compte que la compatibilité au niveau source assembleur suffisait, peu de clients envisageant de garder les mêmes ROM. Un réarrangement des codes instructions machine devenait donc possible, et fut largement mis à profit pour améliorer la régularité du jeu d'instructions.

La technologie utilise le même type de registres que le 6800, donc avec des cellules mémoire dynamiques et une logique combinatoire à séquenceurs dynamiques multiples (rappelons que le 8080, lui, est statique) qui selon les ingénieurs de Motorola présentent l'avantage de se satisfaire de signaux horloges plus simples pour la synchronisation interne que sur un micro statique à états finis, où plusieurs états et fronts d'horloges sont nécessaires là où un seul cycle en dynamique suffit. De même, les timings sont moins critiques. Par contre il y a d'autres inconvénients, et en particulier on a du mal à arrêter le processeur ! Deux lignes de commande permettent de le faire, mais pas complètement : l'une le met dans un état où on reprend périodiquement le contrôle du bus pendant les DMA pour rafraîchir le microprocesseur, et l'autre arrête tout, mais pour une durée maximale de 21 cycles (fréquence horloge 2 MHz, soit le double du 6800).

Une demande importante de la clientèle concernait quatre exigences en matière de programmation, destinées à faciliter l'utilisation de ROM en provenance de fournisseurs différents, les adresses d'implantation de programmes et de ports d'entrée - sortie étant variables. Ces exigences sont les suivantes :

— des programmes insensibles à leur emplacement,
— des variables temporaires sur la pile,
— des opérations indirectes à travers la pile pour les entrées/sorties,
— des opérations indirectes absolues pour les tables de branchement système.

Une analyse des proportions des divers types d'instructions utilisés ne donna pas de résultats très différents de ce que l'on trouve par les ordinateurs classiques : 38 % de chargements et déchar-

gements, 13 % d'appels et retours de sous-programmes, 11 % de branchements conditionnels, 6,5 % de branchements inconditionnels, à peu près autant de comparaisons, autant d'incrémentations, et, à l'habitude, 2,8 % d'additions et de soustractions... Il s'agit ici d'une analyse statique (lignes écrites) et non dynamique (lignes exécutées).

A la suite de tout ceci, le 6809 reçut en particulier les extensions suivantes par rapport au 6800 :

— arithmétique courante sur 16 bits,
— architecture registres plus étendues : 4 de 8 bits et 5 de 16 bits,
— 4 des 16 bits servent d'index ou de pointeur,
— les modes d'adressage indexé sont largement étendus, avec en particulier des champs de 5 ou 8 ou 16 bits signés,
— incrémentation et décrémentation automatique par 1 ou par 2,
— branchement relatif à 16 bits (pour l'indépendance d'implantation),
— généralisation des instructions de fort usage (TFR R1, R2 qui transfère entre registres de toutes tailles avec un seul mnémonique couvrant 42 formes valides, etc.),
— push et pull multiples d'ensembles de registres,
— addition de deux autres niveaux d'interruption logicielle au niveau unique... monopolisé par le logiciel Motorola,
— etc, etc.

L'ensemble de ces additions fait du 6809 un micro très sympathique à programmer, comblant la plupart des déficiences des 8 bits de la génération précédente, chez Mototola et ailleurs, tout en conservant un coût de matériel nettement inférieur à un 16 bit. Il y a certainement bon nombre d'applications justiciables de micros de ce genre.

9 - 7 - 2. Le 8088 : la pintade est un animal élégant

Esthétiquement, le contraste de son cou fin et de sa petite tête par rapport à un corps dodu, mais fuselé, nous a toujours paru intéressant. Elle semble obtenue par étirage local à partir d'un animal plus lourd.

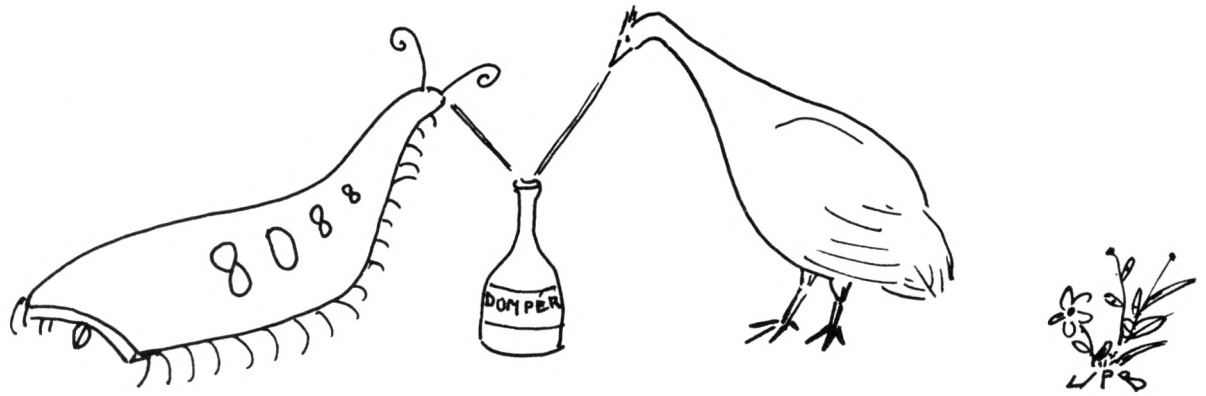
C'est exactement la démarche (jadis suivie par Texas) qu'ont adoptée les ingénieurs de Intel : satisfaits de la puissance et de la souplesse du 8086, ils en ont remodelé une version spéciale à col étroit, le 8088.

Intérieurement, le 8088 est un 8086 :
il n'en diffère que par des détails mineurs (par exemple la queue... qui est plus courte : 4 octets). Les registres sont identiques, généraux ou de segments. L'adressage monte donc aussi au méga-octet. Toutes les instructions produisent le même résultat (quoiqu'avec un nombre de cycles, et un temps de déroulement, qui ne sont pas les mêmes.

La grande différence est que le bus externe de données est large de 8 bits seulement, ce qui :

— permet de concevoir des systèmes plus économiques, et cependant rapides (70 % environ du 8086),
— conserve une compatibilité matérielle directe avec les programmes et les circuits périphériques des 8080/8085,
— offre une compatibilité logicielle avec le 8086.

L'un dans l'autre, c'est un joli coup de rétro, surtout que le coût du microprocesseur est en général négligeable par rapport aux mémoires, aux circuits imprimés, à l'alimentation, au coffret et à tout le reste... La tentation de standardiser à une logique interne identique tous les produits 8 et 16 bits au sein d'un bureau d'ingénierie électronique, d'un constructeur de micro-ordinateurs ou de modules, devient forte !



Convergences...

9 - 8. L'HORIZON PREVISIBLE

Nous voici donc (temporairement tout du moins), au bout de notre périple électronico-informatique. La densité des circuits intégrés double toujours à peu près tous les deux ans, et sauf accident de parcours, que diable va-t-il se passer ?

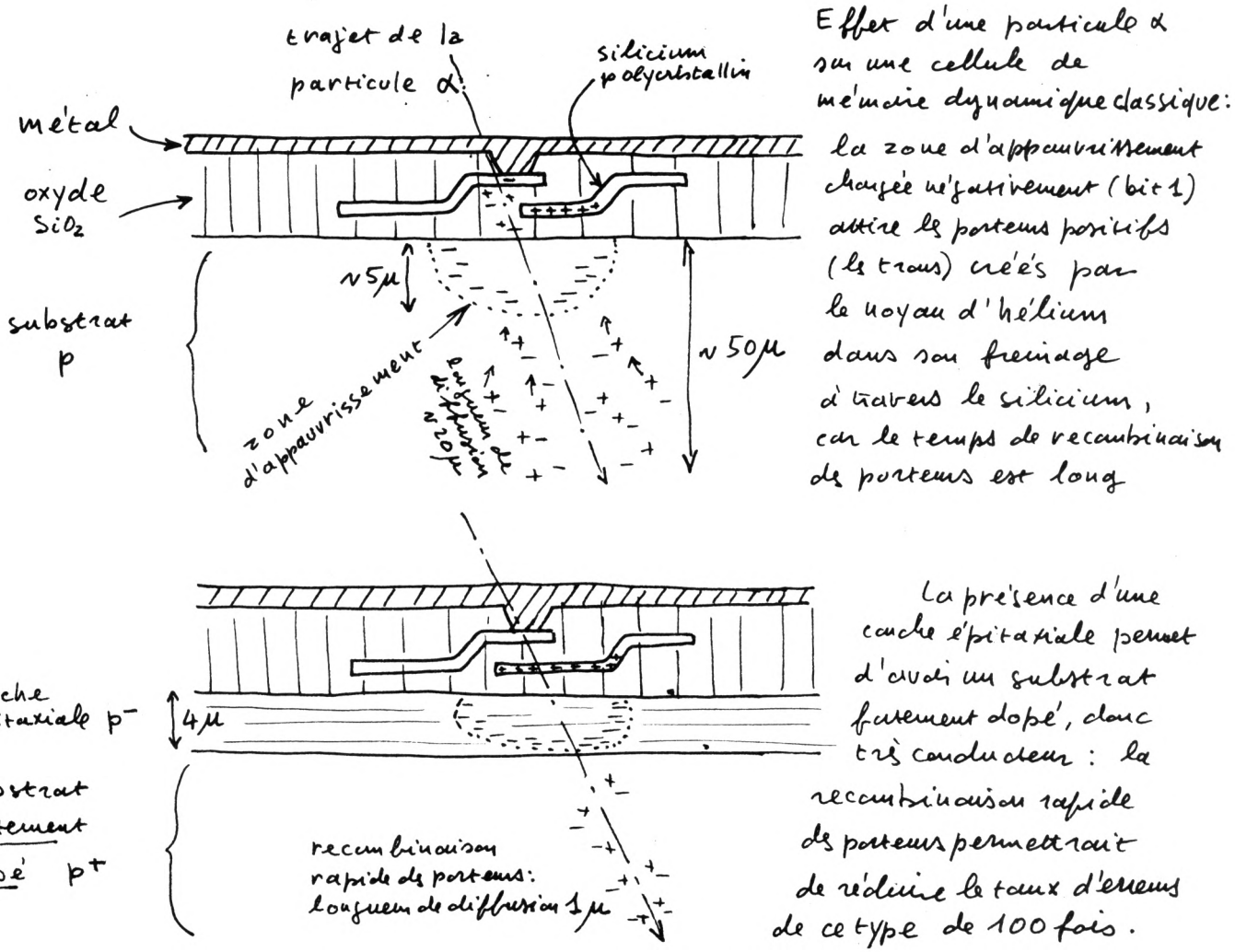
9 - 8 - 1. Mémoires et particules alpha : un défi difficile

Le premier accident de parcours sérieux vient de se produire. De tout temps on savait les mémoires dynamiques (4-2 Statique ou dynamique sujettes à mystérieux problème. Ce sont tout d'abord les rayons cosmiques qui ont été suspectés ! L'un des principaux fabricants de circuits intégrés a acquis 30 tonnes de plomb pour réaliser une chambre d'expérience... et fort heureusement cet abri n'a rien changé aux taux d'erreurs. Fort heureusement, car si les rayons cosmiques avaient été les vrais coupables, il ne restait plus qu'à placer les ordinateurs au fond de galeries de mines désaffectées à plus de mille mètres sous terre ! Depuis peu, on connaît enfin un des mécanismes provoquant des erreurs aléatoires : la radioactivité omniprésente.

Eh bien oui : les céramiques et plastiques emballant les puces pour en faire des boîtiers, comme tout produit, contiennent des traces d'impuretés. Parmi ces impuretés, deux sont dangereuses : l'uranium et le thorium, même en proportions se mesurant en millionièmes. Radioactifs, leurs atomes crachent de temps à autre une particule alpha (un noyau d'hélium, si vous préférez : deux protons plus deux neutrons) de bonne énergie, de l'ordre de 5 méga-électron-volts. Quand elle traverse le silicium de la puce, elle est freinée sur un trajet court de 25 microns environ, et l'énergie dégagée produit 1 à 1,5 million de paires électrons/trous dans le semiconducteur.

Ce qui va se passer alors dépend du type de mémoire, et de l'endroit où la particule a frappé.

Si la mémoire est de type dynamique et que la particule frappe près du transistor/condensateur



Effet d'une particule α sur une cellule de mémoire dynamique classique : la zone d'appauvrissement chargée négativement (bit 1) attire les porteurs positifs (les trous) créés par le noyau d'hélium dans son freinage à travers le silicium, car le temps de recombinaison des porteurs est long

La présence d'une couche épitaxiale permet d'avoir un substrat fortement dopé, donc très conducteur : la recombinaison rapide des porteurs permettrait de réduire le taux d'erreurs de ce type de 100 fois.

de stockage, la charge électrique représentant l'information peut être neutralisée, et un 1 changé en 0. Dans une mémoire statique, il n'y a pas de charge stockée : c'est du courant qui passe en permanence, et rien ne se produit à l'arrivée de la particule. Avantage décisif pour la mémoire statique.

Si la particule alpha frappe près d'une ligne de lecture et qu'il s'agit de mémoire dynamique, il y aura aussi une erreur : on pense à l'heure actuelle, que c'est ce type d'erreur qui est de loin le plus fréquent. S'il s'agit de mémoire statique, le cas est plus complexe. Dans la majorité des mémoires statiques, qui ont une cellule de stockage à 6 transistors, il ne se passera rien. Par contre, pour certaines mémoires statiques, afin de diminuer la consommation électrique, on remplacé deux des transistors par des résistances de charge en silicium polycristallin : si ces résistances de charge ont une valeur trop élevée (certaines font plus d'un gigohm, ne laissant passer que quelques pico-ampères !), il y aura une erreur. Si par contre elles font moins de un mégohm (mémoires ultra-rapides) ou 50 mégohms (mémoires à vitesse normale à faible consommation), il semble que tout se passe bien en général. Mais sur les statiques ultra-rapides (55 ns), il y a aussi un problème : pour atteindre cette vitesse, on a dû dangereusement réduire l'excursion de tension des flip-flops, d'où des marges en tension trop faibles... Finalement, seules les bonnes vieilles statiques à quelques centaines de nanos et à solide consommation échappent à tout ennui !

Pour donner une idée de l'ampleur du problème, avec des céramiques de mauvaise qualité très radioactives et des mémoires dynamiques de 16 K, on peut avoir deux chocs alpha par cellule par cycle de rafraîchissement, et à force de tirer sur la cible à une telle cadence, on aura pas mal de coups au but ! Disons d'ailleurs au passage que, de notoriété publique, les céramiques japonaises sont moins radioactives que les américaines.

Les axes de recherche pour trouver un remède dans l'avenir sont vers des céramiques les moins radioactives possibles, des couches épitaxiales supplémentaires de protection fortement dopées provoquant une recombinaison des paires électrons/trous sur un trajet extrêmement court (un micron) une charge suffisamment élevée dans les cellules mémoire, des lignes de lectures repliées annulant les perturbations communes, des pistes métalliques de protection par-dessus les lignes de lecture, un rétrécissement temporel de la fenêtre de lecture, et si tout cela ne suffit pas, des systèmes auto-correcteurs sur la puce elle-même...

On voit combien nos inquiétudes (exposées en 4 - 2 étaient justifiées !

Nous avons aussi de la mémoire dynamique dans nos modules Alcyane, car le coût de la statique pour les mémoires de grande taille des 16 bits devient peu tolérable, mais avec un contrôle par bit de parité.

9 - 8 - 2. Considérations de conception pour le proche avenir

Devant l'effervescence de la micro-électronique, il y a un peu de quoi se sentir perdu. Le seul capital qui semble garder de la valeur, c'est l'expérience, mais encore à condition de l'entretenir !

Il est très facile de s'enthousiasmer pour de nouveaux produits apparemment très performants. Mais avant de le faire, repensez à ceci : le "tout nouveau, tout beau" s'applique aussi en électronique. Bien que l'obsolescence des produits y soit plus rapide que partout ailleurs, il ne faut pas tomber dans l'excès au plan de l'évolution de la conception. Un microprocesseur ayant quelques années d'existence, en contre-partie d'être moins à la mode, et peut-être un peu moins performant, présente de nombreux avantages :

— il est mieux connu, mieux "supporté", tant par la documentation que par des boîtiers périphériques intéressants,
— il est plus facilement livrable, avec en général plusieurs "secondes sources", signe du succès, il est moins cher, car les secondes sources sont en concurrence
— il est plus simple à utiliser : cette remarque venant du fait que les 16 bits récents sont assez terriblement complexes, et poseront de rudes problèmes aux concepteurs qui aborderont à travers eux les microprocesseurs pour la première fois,
— enfin, il y a moins de "bugs" dans la documentation... voire dans le microprocesseur et les circuits de support eux-mêmes !

Au fil des années qu'a duré cette chronique du microprocesseur, nous avons rencontré un grand nombre d'enthousiastes de la micro-électronique : nous les avons toujours encouragés du mieux que nous avons pu, mais à de nombreuses reprises nous avons aussi dû tirer le signal d'alarme.

Expliquons-nous. A la lecture de certains articles ou revues quasi grand public, beaucoup d'amateurs se sont persuadés que la conception des micro-ordinateurs était une affaire relativement facile.

Ce n'est pas vrai, et nous voulons le signaler une fois de plus avec force.

Faire un montage personnel qu'on utilisera pour soi-même, en toute connaissance de cause, c'est très bien. Réaliser un produit de classe commercialisable, c'est tout autre chose : il faut l'industrialiser, c'est-à-dire le mettre en état d'être fabriqué, et il faut de plus que ce soit un bon produit, sans cela il n'aura aucune chance de réussite.

Les chausses-trappes en matière de conception

électronique sont nombreuses. La logique doit être vue et revue, passée au peigne fin. Les analyses de dispersion de temps de propagation sont à faire dans le détail. Des glitches sournois se glissent subrepticement, et sont parfois très longs à découvrir. Faire des lectures valables à l'oscillo sur des signaux rapides demande une bonne expérience.

L'un des grands pièges est que les circuits logiques n'ont pas toujours un comportement purement logique. Si un signal entaché de bruit entre dans une porte, tantôt la sortie pourra être propre, tantôt pas : il ne faut pas oublier que les fonctions logiques, à l'intérieur du circuit, sont réalisées par de l'électronique, pas toujours identique d'ailleurs pour un circuit de même type d'un fabricant à un autre. Il y a là une source de problèmes particulièrement retors.

Outre les erreurs proprement dites de conception, entre en jeu la qualité du produit. Nous avons parlé ailleurs de la conception générale (modularité, choix des périphériques, logiciel de base), aussi nous concentrerons-nous ici sur un autre problème : celui des erreurs.

Une erreur de temps à autre au cours d'une série de calculs techniques, c'est désagréable, mais à la rigueur tolérable. On s'en aperçoit en général de suite : un point qui "ne file pas" avec les autres, ou le programme se met à dérailler, et on relance. En contrôle de processus industriel, la plupart des micros sont très à l'abri car ils ont la totalité des programmes en ROM, et une alimentation soigneusement filtrée. Mais en gestion, c'est autre chose : quand la comptabilité ne balance pas, ou qu'un programme éclate au beau milieu d'une centralisation comptable, on fait grise mine. En matière de matériel de gestion il y a beaucoup d'appelés, mais peu d'élus.

9 - 8 - 3. Les sources d'erreurs dans un ordinateur

Les sources d'erreurs dans un ordinateur sont multiples, surtout lorsqu'on tient compte des périphériques. En nous bornant à l'unité centrale, il y a quatre causes principales d'erreurs aléatoires pendant le fonctionnement de l'ordinateur :

— *le bruit* : sur chaque carte de l'ordinateur, des centaines ou des milliers de circuits passent leur temps à commuter à grande vitesse, générant un bruit de fond important. La marge d'immunité au bruit des boîtiers utilisés étant ce qu'elle est, une pointe de bruit ultra-courte peut la dépasser et provoquer une erreur par-ci, une erreur par-là. Un travail patient est nécessaire pour réduire le niveau de bruit le plus possible ; il commence dès la conception (disposition du bus, des lignes, des masses électriques, mécaniques, etc.) et se continue par la suite (nature et disposition des découplages, changement de composants, etc.).

Rentrent également dans cette catégorie les bruits hertziens externes ou internes à ausculter soigneusement sur les cartes, le bruit de fond électro-magnétique en provenance de l'alimentation (les transfos en double C donnent des résultats remarquables). De bons coffrets métalliques sont la meilleure protection contre les agressions externes. Les coffrets plastiques "design" laissent tout passer ;

— *les fluctuations d'alimentation* : le 220 volts secteur est... ce qu'il est, selon l'heure, l'emplacement géographique (ville, campagne, zones industrielles, près du transfo, en bout de ligne, etc.) et même la météo (orages). Il peut être sujet à microcoupures, et véhicule à peu près partout des parasites à front raide de sévérité plus ou moins grande dûs aux tubes fluorescents, machines à écrire, sonnettes, etc. Il est impératif que l'ordinateur se comporte bien en face de ce genre de perturbations, et ça ne se fait pas tout seul ! Ce seul point peut faire toute la différence entre une machine utilisable professionnellement et un simple gadget.

— *l'électricité statique* : c'est un fléau dû aux moquettes synthétiques, blouses nylon, pantalons tergal, semelles de caoutchouc... et chauffage central. Là encore la nature du coffret et les connexions de masse sont très importantes, mais il y a bien d'autres points. Par exemple, certains claviers à touches sans contact mécanique sont très bien sur le papier en ce qui concerne la fiabilité théorique, mais sont de véritables catastrophes en face des problèmes pratiques d'électricité statique ;

— *le rayonnement alpha* : ce phénomène, que nous avons décrit plus haut en détail, frappe tous les micro-ordinateurs utilisant de la mémoire dynamique sans contrôle de parité. Ce qui veut tout simplement dire que 90 % des micro-ordinateurs du marché sont assez peu recommandables pour la gestion...

Nous avons parfaitement conscience de faire éclater une bombe en écrivant ces lignes. Ayant tiré la sonnette d'alarme dans L'Informatique Nouvelle en septembre 1976, nous trouvons quand même assez extraordinaire qu'il ait fallu attendre l'été 78 pour que le premier article à ce sujet paraisse (dans la plus importante revue d'électronique des Etats-Unis), et que le pot-aux-roses n'éclate vraiment qu'en 79 avec - tenez-vous bien - plus d'une demi-douzaine de communications sur le problème au symposium international, tenu en avril à San Francisco, sur la Physique de la Fiabilité !

Pour en revenir à la conception en général, on ne peut guère télescoper les étapes, ou alors on s'expose à un ou des rappels généraux du maté-

riel en clientèle, qui terniront simplement l'image si c'est un gros constructeur, ou le mettront en faillite si c'est un petit. Il n'y a vraiment que dans le domaine des amateurs que n'importe quoi passe, car c'est finalement ce qu'ils veulent, mettre la main à la pâte pour apprendre et avoir le plaisir de dompter leur électronique. Nul constructeur n'est à l'abri de grosses catastrophes, pas même IBM.

Je me rappelle parfaitement mon premier coup d'œil dans un 360/30. C'était un des tout premiers livrés. A l'intérieur de l'unité centrale, vision extraordinaire : un enchevêtrement de petits tuyaux d'air comprimé, un manomètre, un réservoir d'air, un compresseur, que sais-je !

La microprogrammation était stockée sur cartes perforées normales 80 colonnes : ce n'est pas une plaisanterie, le système était très astucieux, trop même. Les cartes perforées avaient reçu une déposition de pistes et de plots rectangulaires en encre métallisée, et on les passait dans une repro pour les perforer selon le jeu maître. Là où un zéro devait figurer, une perforation enlevait le plot métallisé (l'inverse donc de la carte perforée classique). Là où le plot restait, c'était un 1. Les cartes étaient ensuite mises par paires dans les cases de l'unité à microprogrammer, séparées par un petit sac qu'on gonflait d'air (d'où le compresseur et les tuyaux !). Le sac appuyait fermement les cartes sur leur panneau à lecture capacitive.

Dix huit mois plus tard, à l'occasion d'une autre mission de conseil, j'ouvre à nouveau les portes de la même unité centrale de 360/30 : stupéfaction, plus aucun rapport ! Plus de tuyaux, plus de compresseur, seulement des fils électriques. La technologie avait été entièrement changée, tant le système précédent devait être source d'ennuis en clientèle (sensibilité aux parasites ? Le bruit courait que le passage d'un opérateur en blouse de nylon roulant un chariot de cartes devant la machine suffisait à produire des erreurs... vrai ou faux, il y avait comme un problème). Le 360/40 marchait, lui, parfaitement, et si mes souvenirs sont bons c'est sa technologie à lecture inductive (avec le fameux "hareng") qui avait dû être transférée sur le 360/30. On trouverait bien d'autres exemples récents.

La vie d'un ordinateur, depuis sa naissance dans le cerveau des concepteurs, passe par toute une série d'étapes qui sont en gros les suivantes :

— *prototype*
- projet logique
- construction du prototype (ou des prototypes)
- mise au point du prototype
- modifications du prototype
- premier logiciel rudimentaire

— *industrialisation*
- implantation des cartes
- dessin des mylars
- étude et réalisation de l'alimentation
- organisation et mécanique du coffret
- connectique
- réalisation d'un logiciel de base utilisable

— *vie du produit*
- étude et réalisation de nouveaux interfaces
- nouvelles versions d'interfaces anciens
- mise à niveau technique suivant les rapports d'entretien en clientèle, avec effet rétroactif sur le matériel installé
- adaptation du logiciel de base aux nouveaux interfaces
- développement du logiciel de base en versions successives ayant de plus en plus de possibilités (tri, séquentiel indexé, traitement de texte, etc.)
- extensions du système de base (nouvelles architectures, telle l'architecture NODALE, extension à de nouveaux microprocesseurs, de nouveaux types de mémoire, etc.).

La durée de ces étapes dépend beaucoup de l'ampleur du système : mettons un an pour la première, un an et plus pour la seconde, et pas mal d'années pour la troisième (nous restons dans le domaine des micros, pour les ordinateurs lourds le cycle est bien plus long).

9 - 8 - 4. Le proche avenir

Que nous réserve le proche avenir pour tout ceci ? Beaucoup de travail, mais aussi beaucoup d'opportunités. La nouvelle informatique vient par vagues. Celle des minis a commencé à déferler au-dessous des ordinateurs lourds... puis le niveau est monté rapidement ! Un DEC-20 ou un VAX-11/780 avec adressage virtuel de quatre milliards d'octets sont de bien grosses machines.

Celle des micros a commencé à déferler au-dessous des minis... et le niveau monte plus rapidement encore ! Nous avons vu les performances des nouveaux 16 bits, elles permettent la conception de matériels qui mordent sérieusement sur les plate-bandes des minis. En équivalent instructions ordinateur lourd, une machine conçue autour d'un 8086 (pour ne citer que celui-là) tournera entre 50 et 100.000 instructions par seconde. A titre de comparaison, un 360/40 tournait entre 30 et 60 000 i/s, et un 360/50 entre 100 et 180 000 i/s. La taille mémoire maxi du 40 était très inférieure à celle de notre "micro". Le canal d'accès mémoire du 40 débitait 2 octets en 2,5 microsecondes, soit 0,8 mégaoctets/seconde (le DMA d'Alcyane est déjà un tout petit peu plus rapide), et celui du 50 débitait 4 octets en 2 microsecondes, soit un peu moins que celui du 16 bits.

Il ne faut cependant pas prétendre que les performances globales des nouveaux micro-ordinateurs seront équivalentes à celles d'un milieu de gamme 360 : un ordinateur lourd possède entre autres des canaux, c'est-à-dire des contrôleurs

de périphériques très évolués, qui lui assurent des débits moyens d'entrée/sortie très supérieurs à ce qu'on peut atteindre avec des structures plus simples. Mais c'est le rapport performances/prix qui compte, et là la comparaison est écrasante.

Par ailleurs, le coût d'un boîtier 8 ou 16 bits est suffisamment faible pour qu'on puisse en mettre plusieurs par système : par exemple, la dernière carte interface Alcyane développée comprend deux microprocesseurs, deux UART (convertisseurs série/parallèle pour transmission/réception) et deux bonnes mémoires tampon. Elle effectue une gestion complète de transmission, jusqu'à 1 km sans modem et sur réseau avec modem, déchargeant complètement le processeur de cette tâche, ce qui permet des systèmes multi-écrans/claviers à distance comme sur minis : une telle carte est en fait ni plus ou moins un canal.

Enfin, de nouvelles unités de disques arrivent. La gamme Cynthia de Cii-HB, pour laquelle nous avons réalisé le contrôleur formateur mi-78, a débuté par le D 120 de 10 Mo (voir Small is beautiful, again : cavalier seul), se poursuit par le D 140 à 10 Mo fixe plus 10 Mo amovible, puis le D 160 à 120 Mo fixe, ce qui est une fort belle capacité. La combinaison D 160 plus D 120 pour sauvegarde est très intéressante, et étend considérablement les possibilités du système.

Pour le proche avenir, deux types de disques paraissent prometteurs pour les systèmes moyens : les microdisques rigides 8 pouces fixes dans le même encombrement qu'une unité à disque souple, et des disques souples à grande vitesse à têtes volantes, tous deux à des prix très compétitifs pour des capacités de 2 à 10 millions d'octets environ.

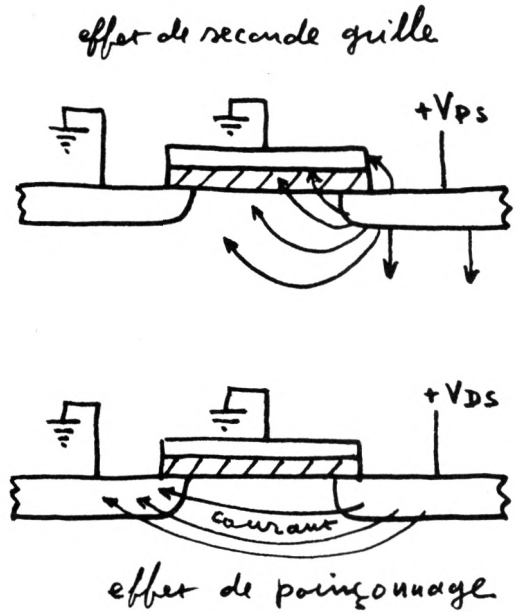
La vague des micros monte donc sans cesse. Peut-on encore les appeler micros ? Cela n'a plus guère de sens. Ce sont tout simplement des ordinateurs.

L'acquit de logiciel pour les fabricants va devenir un capital de plus en plus précieux. Le plus difficile sera de limiter sa complexité par de bons choix, de façon à ne pas retomber dans le vieux piège du système d'exploitation complexe...

Le doublement incessant de densité des composants nous entraîne de façon véritablement vertigineuse, comme dans l'histoire des grains de blé sur l'échiquier. Le premier microprocesseur, le 4004, a vu le jour en 1971. Sept ans après, le même fabricant sort le 8086 qui a 10 fois plus (et davantage) de composants sur sa puce, calcule 200 fois plus vite et possède une mémoire centrale 250 fois plus grande : d'autres microprocesseurs encore plus puissants vont suivre ! Où va-t-on s'arrêter ?

La limite ultime est encore assez loin, mais nombre de paramètres devront être modifiés pour l'atteindre. Dans la technologie actuelle, les surfaces de puce maximales pour un rendement convenable en production ne doivent pas dépasser 25 à 35 millimètres carrés. Une amélioration régulière de la qualité des cristaux et des traitements permettra d'aller progressivement au-delà. La finesse des lignes décroît constamment. L'étroitesse des canaux et la minceur des oxydes de grille augmente à tension égale l'intensité du champ électrique et change sa forme, ce qui fait apparaître des effets de second ordre et imposera petit à petit des changements dans les habitudes :

— *effet de seconde grille.* Le champ électrique de la jonction du drain arrive sous l'oxyde de grille et perturbe le fonctionnement, augmentant le courant de fuite si la tension augmente au-delà d'un seuil... qui s'abaisse au fur et à mesure que le canal devienne plus étroit. Lorsque le champ s'étend jusqu'à la jonction de la source, les phénomènes s'aggravent ;



— *réduction de tension d'alimentation.* Elle accroît nettement la fiabilité des structures fines, par réduction de la valeur des champs électriques. De plus, les performances s'accroissent nettement : pour le H-MOS à deux microns de 1980, une tension d'alimentation comprise entre 2 et 4 volts diminue de moitié le délai de propagation par porte, et réduit de 60 % la puissance dissipée. N'oublions pas la limite pratique de 1 watt par boîtier pour que l'électronique reste fiable ;

— *nouveaux boîtiers.* Devant l'augmentation constante du nombre de composants par puce, il va falloir passer à autre chose que le boîtier classique DIL/DIP. L'un des candidats, présenté par Intel et 3M, est le QUIP (quad in line package). Comme son nom l'indique, il comporte 4 rangées de 16 broches, ce qui le rend à peu près deux fois moins encombrant qu'un boîtier DIL

64 broches. Sa conductibilité thermique est bien meilleure (35° C/W contre 60 à 90) : pouvant dissiper près de deux watts, il permettra des puces de 50 à 100 000 transistors. Les avantages secondaires sont nombreux : insertion plus facile, détrompeur incorporé, points de test plus accessibles, boîtier moins coûteux, capacités entre broches plus faibles, etc.

9 - 8 - 5. Etat de l'art 1982

La deuxième édition de cet ouvrage nous donne l'occasion de voir comment, en deux années, ont évolué grosses et petites machines, microprocesseurs et puces mémoire. Comme nous l'avions laissé pressentir, le système ne s'est pas ralenti, sauf en ce qui concerne les mémoires dynamiques à la suite du problème des particules alpha, mais même là nous allons voir les stupéfiants développements en cours, qui font plus que rattraper le temps perdu !

Ordinateurs géants

Nous avons pris comme exemple le Cray-I et sa technologie : or justement Seymour Cray vient de dévoiler les grandes lignes de son nouveau super-ordinateur, le Cray-II. Une fois de plus, il s'agira bien entendu de la machine la plus puissante du monde : six fois plus rapide que le Cray-I en calcul normal, et douze fois plus en calcul vectoriel.

Ce qui frappe de prime abord, c'est la compacité remarquable de la machine, qui comme la précédente a la forme d'un cylindre, mais avec cette fois une hauteur de 65 centimètres et un diamètre de 96 centimètres seulement. Tout ceci bien que la mémoire centrale soit huit fois plus importante, passant à 32 millions de mots, et que l'unité centrale soit un véritable multiprocesseur à quatre processeurs, et non un monoproceseur plus des unités fonctionnelles ne méritant que le qualificatif de processeur parallèle.

On s'en doute, une nouvelle technologie est nécessaire, aussi bien pour la disposition des circuits intégrés que pour le refroidissement.

En ce qui concerne les boîtiers, toujours en ECL fabriqués par Fairchild, ils témoignent d'une évolution continue. Une projection directe du masque sur la tranche avec répétition a permis d'atteindre une géométrie de deux microns, d'où des puces mémoire quatre fois plus denses atteignant 16 k et des puces logiques à 300 ou 500 picosecondes passant de deux à seize portes. Le cycle de la machine est de quatre nanosecondes.

La disposition des cartes est totalement différente de ce qu'on connaissait. Les modules 3-D regroupent huit cartes par module au lieu d'une seule sur une épaisseur de deux centimètres et demi, et les modules sont distants de trois millimètres seulement. Or, la dissipation globale semble légèrement supérieure à celle du modèle précédent qui faisait, rappelons-le, 1,8 m de haut sur 1,2 m de diamètre !

On s'en doute, le mode de refroidissement est nouveau, c'est l'immersion totale. Le liquide, absolument neutre, est celui utilisé pour remplacer le sang dans certains cas de transfusion. Tout ceci signifie qu'il faut vidanger l'unité centrale pour remplacer une carte, c'est le coût pratique de cette technologie.

Ordinateurs de poche

Spectaculaire témoignage des progrès continus et rapides de la micro-électronique, cinq ou six modèles d'ordinateurs de poche sont arrivés sur le marché. Nous allons en décrire rapidement deux, de conception différente : le premier vise principalement le marché des amateurs et des utilisateurs scientifiques, le second celui des utilisateurs professionnels, de gestion entre autres. Tous deux ont une apparence extérieure voisine.

Le Radio-Shack TRS-80 Pocket Computer, vendu 250 dollars, fabriqué par Sharp, doté d'un clavier et d'un affichage (à cristal liquide) alphanumériques, possède un interprêteur Basic étonnamment complet. Il utilise deux microprocesseurs à 4 bits en C-MOS, à très faible consomation, à ROM incorporée. Le premier gère le clavier, organise les données, effectue le parsing des opérations arithmétiques et la reconnaissance de la syntaxe des expressions Basic, et loge les codes correspondants dans une mémoire tampon. Le second exécute les instructions et gère l'affichage. On peut loger des programmes allant jusqu'à 250 lignes Basic !

Les Panasonic et Quasar (identiques à des détails près), résultent d'une collaboration entre Friends Amis (de San Francisco, très représentative de la crème de Silicon Valley) et Matsushita. Leur HHC (hand held computer, ordinateur qu'on peut tenir à la main), vendu 400 dollars, exécute directement en microcode du SNAP, langage de type FORTH. Le microprocesseur est un 6502, donc à 8 bits. Des « capsules » de ROM enfichables de 2 à 16 K octets permettent l'accès soit à des données (traduction Français/Anglais, etc.), soit à des programmes (jeux, etc.), soit à des langages (capsule Basic). Des schémas élaborés de compression de données sont utilisés de façon interne (codes de Huffman, codons). Au nombre des accessoires existants ou en développement on note : modem acoustique, extension mémoire vive, interfaces TV couleur et cassettes, imprimante, etc.

Micro-ordinateurs 16 bits

Ils commencent à apparaître sur le marché, mais encore en nombre faible par rapport aux 8 bits dont les performances sont suffisantes pour une grande partie des applications, et le prix inférieur.

L'entrée la plus notable est celle d'IBM avec son « Personal Computer », construit autour du 8088, ce qui — comme nous l'avons noté dans le paragraphe sur ce microprocesseur — nous paraît un choix intelligent.

Microprocesseur 32 bits

Le nouveau microprocesseur à 32 bits Intel iAPX 432 promet une véritable révolution sur les marchés de l'informatique. Il va permettre en effet pour la première fois la réalisation d'unités centrales de micro-ordinateurs d'une

puissance largement équivalente à celle des ordinateurs classiques. Qu'on en juge plutôt par les comparaisons ci-après, indiquées par Intel, et portant sur des unités centrales (à l'exclusion de tout périphérique).

Un IBM 370/158 a un volume de 16.000 dcm³, un prix de 300.000 dollars, et fonctionne à 2.000.000 d'instructions par seconde.

Un VAX 11/780 de DEC a un volume de 5.000 dcm³, un prix de 100.000 dollars, et fonctionne à 1.000.000 d'instructions par seconde.

Le iAPX 432 va permettre de réaliser des unités centrales d'un volume de 55 dcm³, coûtant 5.000 dollars, et fonctionnant à une vitesse qui, selon le nombre de microprocesseurs utilisés, ira de 500.000 à 2.000.000 d'instructions par seconde.

L'une des originalités du système est que le noyau primaire du système d'exploitation est stocké sur silicium. Les « mécanismes » fondamentaux correspondants permettent la répartition automatique des traitements sur les processeurs multiples en fonction de paramètres concernant la priorité, l'heure maximale de sortie et les affectations de tranches temporelles, de même que les affectations mémoire paramétrées.

Les « politiques » constituant la superstructure du système d'exploitation restent sous forme classique de logiciel : optimisation des charges de traitement au temps réel, temps partagé, par lots, ou dynamique, frontière maximale des affectations mémoire, etc.

On remarque donc que toute la communication interprocessus et la synchronisation sont figées sur silicium, de même que les règles de base des affectations des ressources de traitement ou de mémoire.

Le système 432 comprend trois circuits intégrés : le 43201 décode les instructions, le 43202 les exécute, et le 43203 constitue le système d'interface des entrées/sorties. L'ensemble de ces trois circuits intrégrés comprend plus de 200.000 composants. L'architecture utilisée permet si on double les circuits une vérification du traitement par redondance fonctionnelle détectant les défaillances matérielles. Les longueurs d'adresses sont de 24 bits, permettant l'adressage d'un espace physique de 16 millions d'octets, et des adresses physiques de 40 bits permettant d'adresser un espace virtuel de 1 000 milliards d'octets. Les types de données traitées sont de 8 bits, 16, 32, 64 et 80 bits. Il existe des opérations à trois opérandes. La mémoire cache est l'une des caractéristiques d'architecture permettant d'expliquer la vitesse élevée obtenue (rappelons que le 8086 à 16 bits utilise déjà une petite mémoire cache incorporée dans sa puce).

L'assembleur du 432 est difficile à utiliser (ses instructions vont de 6 à 300 bits de long). L'architecture du 432 est entièrement orientée vers l'utilisation directe de Ada, ce qui en fait un instrument extrêmement puissant à la pointe du progrès en informatique, mais par-là même certainement pas à mettre entre toutes les mains !

Il s'agit là d'une annonce majeure dans le domaine de la micro-informatique, correspondant à des systèmes qui, compte tenu de la technicité prévisible de leur développement, seront vraisemblablement mis sur le marché vers 1983 ou 84.

L'os en silicium

L'os en calcium soutient l'homme. L'os en silicium soutient l'ordinateur. Deux règnes différents dans la nature. Qu'on nous pardonne le calembour, il n'y a qu'en franglais qu'il est possible, et il est difficile d'y résister...

Etape de pénétration importante de la nouvelle concurrence que constituent les fabricants de composants dans le domaine informatique, la puce complémentaire système d'exploitation (ou plutôt *exécutif*) fait son apparition sur le marché.

Jusqu'ici, le premier exécutif sur silicium faisait deux parties intégrantes d'un microprocesseur (le iAPX 432), mais voici maintenant qu'arrivent les « puces système ».

Le boîtier céramique à 40 broches 80130 de Intel lorsqu'on le couple aux microprocesseurs 8086 ou 8088, ajoute à leur propre jeu d'instructions 35 instructions (primitives) système. Elles peuvent être employées comme base pour la réalisation d'un système d'exploitation « maison », temps réel et multitâches, ou comme soutien de l'IRMX 86 de Intel, tout prêt. Cinq nouveaux types de données systèmes deviennent disponibles :

- de travaux : elles permettent le soutien de la multiprogrammation,
- de tâches : elles permettent de créer, détruire et contrôler l'exécution de tâches en fonction de priorités pré-établies par interruptions,
- de segments : elles sont utilisées pour la gestion mémoire, avec affectation dynamique dans l'espace mémoire centrale de un million d'octets,
- de boîte aux lettres : elles permettent la communication entre les tâches et leur synchronisation,
- de région : elles empêchent une tâche de prendre le contrôle d'une ressource partagée pendant qu'une autre tâche en modifie le contenu, ceci au moyen de « l'exclusion mutuelle ».

En plus du noyau d'OS de 16 k octets, le boîtier contient un contrôleur d'interruption programmable, un temporisateur banalisé et un pour l'OS, un générateur de baud et quelques autres gracieusetés logiques. Ce bloc de 16 Ko est à placer dans le segment primaire d'adressage de 64 Ko, et il faut réserver 2,5 Ko de RAM pour les stockages de vecteurs d'interruptions, piles et variables système diverses.

Une deuxième version du 80130 comprendra un mini-OS complet, le populaire CP/M-86.

Mais voici maintenant que les « indépendants » apparaissent, proposant leurs propres puces système comme compléments des microprocesseurs des fabricants de composants ! Le premier est Hunter & Ready, avec le VRTX, logé sur une ROM de 4 Ko. Il existe déjà pour le Z 8002, le 8086, le 68000 et le 16000 de National Semiconductor.

Il est certain que la présence de ces nouveaux composants système additionnels facilitera l'utilisation des microprocesseurs à 16 bits, déjà passablement complexes, permettant de déployer plus facilement leur puissance dans les systèmes de traitement ou d'automatisation de l'usine ou du bureau, où à notre avis ils domineront pendant une bonne décennie, les 32 bits se cantonnant dans des niches plus spécialisées et dans le secteur des minis.

9 - 8 - 6. La puce de quatre millions de bits

La limite ultime du nombre de composants par puce sur du silicium, prévue de façon nous paraissant pourtant optimiste dans la précédente édition de cet ouvrage, semble d'ores et déjà pulvérisée. A la fois en valeur et en date !

Le problème des particules alpha a reçu plusieurs solutions essentiellement à base d'un film de protection en polyimide et de redondance. De toutes façons, il est absolument nécessaire d'utiliser bit de parité ou code autocorrecteur avec les moindres dynamiques, qui maintenant dominent totalement le domaine des puces mémoire de forte capacité en raison de leur grande densité et de leur faible dissipation. Déjà les puces 64 K commencent à apparaître sur des produits commerciaux, et les 256 K sont fabriquées au laboratoire.

Mais la bombe, et ce sera la conclusion de notre ouvrage, c'est la puce de quatre millions de bits !

De nouveaux détails émergent sur la genèse de cette puce mémoire qu'on aurait normalement seulement attendue pour la fin du siècle.

Le développement en est effectué par Condesin, Inc. (qui vient de quitter le Texas pour Cupertino) pour le compte d'une société californienne fermée de recherche et développement, Ebram, Ltd. Le Docteur Alton O. Christensen, Senior, a fait apport à Condesin de sa technologie sous forme de brevets accordés ou disposés et en cours d'examen : nous rappelons qu'il a été le premier à intégrer des mémoires vives MOS, et que ses brevets originaux sont actuellement propriété de la Shell Oil Co. Il vient d'annoncer que la structure globale de la puce comporte huit rangées de huit pages de 64 K, ayant chacune un quart de millimètre de côté, ce qui avec les circuits de commande donne à la puce la dimension surprenante de 2,5 mm sur 4,2 mm seulement (les puces japonaises expérimentales de un mégabit à redondance sont énormes à côté de celle-ci). Bien que l'inscription directe par faisceau d'électrons soit utilisée pour des circuits de lecture/écriture, la production des puces utilisera des méthodes de photolithogravure normales.

Les licences devraient être disponibles pour les fabricants de composants selon un calendrier permettant l'introduction du produit... l'année fatidique 1984.

On ne sait que penser devant ce développement stupéfiant : a-t-il de sérieuses chances d'aboutir dans les délais ? Tout est là. Si c'est le cas, la donne des cartes sur les marchés électronique et informatique va brutalement changer, et l'évolution prévisible des micros va recevoir un formidable coup d'accélérateur : d'incroyables machines électroniques, parlant, écoutant, répondant, « réfléchissant », sont pour bientôt.

ANNEXE
ECRIVEZ VOTRE PROPRE ASSEMBLEUR*

Si vous avez déjà travaillé sur des micro-ordinateurs, vous avez sans doute vu des programmes écrits en langage d'assemblage. Vous savez probablement que les programmes en langage d'assemblage doivent être traduits en langage machine avant de pouvoir être exécutés par l'ordinateur. La traduction est habituellement faite par un autre programme appelé "assembleur". Le langage d'assemblage vous permet d'écrire des noms mnémoniques (faciles à se rappeler) pour les noms d'instructions et de données, plutôt que des codes binaires, les programmes peuvent ainsi être écrits plus vite et avec un peu moins d'erreurs. L'assembleur fait le travail fastidieux de mettre ensemble, ou d'assembler, toutes les combinaisons correctes de bits pour construire le programme en langage machine.

La plupart des assembleurs actuellement opérationnels pour les micro-ordinateurs sont des assembleurs horizontaux. Ils tournent sur de gros ordinateurs ou sur des systèmes à temps partagé, et produisent des sorties qui doivent être chargées d'une certaine façon dans le système du micro-ordinateur. Les services commerciaux de temps partagé sont coûteux, et l'avantage principal de posséder un ordinateur-maison est d'être capable d'exécuter des tâches, telles que l'assemblage d'un programme, sur votre propre système à un prix extrêmement bas. Sachant qu'un assembleur résident — tournant sur votre propre micro-système — est introuvable ou très coûteux, vous pourriez être intéressé d'écrire votre propre assembleur. En le faisant vous-même vous pouvez apprendre beaucoup sur la programmation et sur l'étude du logiciel aussi bien que sur les spécifications de votre micro-ordinateur, économiser vous-même le prix du développement du programme, et produire un langage modifié adapté à vos besoins et à votre imagination. Et qui sait ? — vous pourriez toujours trouver d'autres personnes, amateurs ou professionnels, qui seraient d'accord pour vous payer pour une copie de ce programme assembleur que vous avez eu tant de plaisir à écrire.

---

* Cet article a été publié en français pour la première fois dans *L'Informatique nouvelle* d'avril 1977.

Maintenant il est normal de vous prévenir que l'écriture d'un assembleur n'est pas une mince affaire. Il vous faudra beaucoup de temps et peut-être quelques mémoires RAM supplémentaires pour recevoir le produit fini. Mais de tels obstacles n'ont jamais arrêté une personne ayant autant d'enthousiasme que vous. Donc la seule question est de savoir comment vous allez vous y prendre pour écrire un assembleur.

Qu'est-ce que fait un assembleur ?

Pour répondre à cette question, nous devons jeter un œil sur quelques instructions machine classiques et sur la façon dont elles pourraient être écrites en langage d'assemblage. Une instruction machine consiste en un code binaire pour quelques opérations, telles que l'addition et en un ou plusieurs nombres binaires représentant les "opérandes" de l'opération. Le nombre binaire pour un opérande peut avoir deux sortes d'interprétations : il peut donner la valeur binaire de, disons, un nombre, ou le code ASCII d'un caractère, où il peut donner l'adresse binaire d'une adresse mémoire qui contient la valeur de l'opérande. Par exemple sur le Motorola 6800 la combinaison de bits :

1000 1011	0011 0000
code-op	opérande

signifie "ajouter le nombre 48 (00110000 en binaire) à l'accumulateur A". Ceci serait représenté en langage d'assemblage par :

ADDA #$ 48

notons combien il est plus pratique d'écrire les choses de cette manière ! Par contre la combinaison de bits :

1001 1011	0011 0000
code-op	opérande

signifie : "ajouter le nombre de 8 bits se trouvant à l'adresse mémoire 48 à l'accumulateur A". Cela pourrait s'écrire en langage d'assemblage :

ADDA BETA

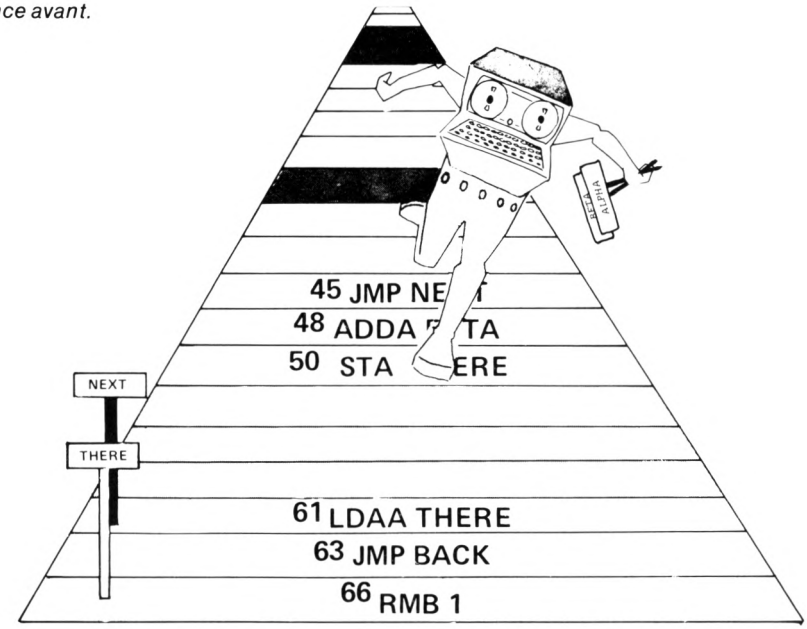
Il arrive que, juste après la dernière instruction d'un programme ayant 48 octets de long, le programmeur ait aussi écrit :

BETA RMB 1

signifiant : "réserver un octet mémoire à cet endroit et l'appeler BETA".

Ces exemples illustrent les fonctions de base d'un assembleur. Dans le premier cas, l'opérande de l'instruction était le nombre à ajouter (on l'appelle souvent "opérande immédiat"). L'assembleur lit les caractères "ADDA" et substitue le code-op binaire correspond 10001011, puis convertit le nombre décimal 48 en 00110000, son équivalent binaire. Dans le second cas, l'opérande était l'adresse d'une adresse mémoire. Le programmeur appelle cette adresse mémoire BETA et décide de la mettre après les instructions d'un sous-programme, qui commence à l'adresse 0. Il l'indique par le mnémonique RMB "réserver octet mémoire" (reserve memory byte). Si l'instruction du langage d'assemblage ne représente pas vraiment une instruction, mais si elle dit à l'assembleur ce qu'il doit faire, elle est souvent appelée une "pseudo-instruction". L'assembleur lit le programme en entier, comptant le nombre d'octets pris par le

Fig. 1. Le problème de la référence avant.



sous-programme, et détermine l'adresse de l'adresse mémoire appelée BETA qui est par conséquent 48, ou 00110000 en binaire. Il assemble cette adresse dans l'instruction.

Tout ceci est bien beau. Mais étant un lecteur alerté vous demandez : "Attendez une minute ! Que se passe-t-il lorsque l'instruction ADDA est au milieu du sous-programme ? Quand l'assembleur lit l'instruction ADDA, il ne sait pas combien d'octets prendra le reste du sous-programme, et de plus il ne sait pas (encore) que l'adresse mémoire BETA est supposée être réservée juste après le sous-programme, puisqu'il n'a pas vu la pseudo RMB".

La figure 1 illustre un problème commun à tous les assembleurs et à tous les compilateurs, souvent appelé "le problème de la référence avant". Il n'y a pas de solution pure. Dans ce cas le programmeur pourrait avoir réservé l'adresse mémoire BETA avant le sous-programme plutôt qu'après. Mais supposons que le sous-programme comprenne une instruction de "saut" ou de "branchement" :

```
                JMP          NEXT
                 •
                 •
                 •
NEXT            ADDA         #7
```

Il est pratiquement impossible d'essayer d'écrire tous ces programmes sans aucun saut en avant !

Il y a généralement deux façons de faire face à ce problème. La première est de lire le programme une fois, mais de garder les sections du programme en mémoire jusqu'à ce que toutes les références avant soient résolues. Si la RAM nous coûte cher dans un système de micro-ordinateur, nous rejetterons cette approche. La seconde alternative est de lire deux fois le programme ; un assembleur qui adopte cette stratégie est appelé assembleur à deux passages. Cette approche est lente, mais aussi bon-marché, et c'est ce que nous voulons !

Lors de la première lecture du programme (c'est-à-dire lors du premier passage), il regarde simplement les mnémoniques des instructions, compte le nombre d'emplacements mémoire que chaque instruction prendra, et construit une table des symboles en mémoire qui liste tous les noms définis par le programmeur pour les emplacements mémoire et leurs adresses correspondantes. (Nous avons besoin de RAM pour cela, mais pas autant que le nécessiterait la première approche). Ce processus est illustré (d'une manière un peu fantaisiste) par la figure 2. Notons que l'assembleur ramasse seulement les étiquettes des instructions, ignorant (dans le cas du premier passage) les apparitions des symboles identiques dans l'ensemble des opérandes des instructions.

Au second passage, les codes opérations binaires sont mis à la place des mnémoniques des instructions, les constantes sont remplacées par leur représentation binaire, et les noms définis par le programmeur sont remplacés par leurs adresses mémoire, trouvées dans la table des symboles. Ceci est illustré figure 3. Chaque nom se trouvant dans la zone opérande d'une instruction, qui n'est pas encore sur la table des symboles au deuxième passage, est indéfini dans le programme, et conserve un message d'erreur. Une autre remarque : regarder le code-op binaire d'un mnémonique d'une instruction est essentiellement la même chose que de regarder l'adresse d'un nom défini par le programmeur, donc la table des symboles peut être utilisée dans les deux cas.

Il devrait être assez clair maintenant qu'un assembleur passe la plupart de son temps :

— à analyser les caractères, à rechercher les noms, les nombres et les symboles de ponctuation et,

Fig. 2. Premier passage ramassage des étiquettes.

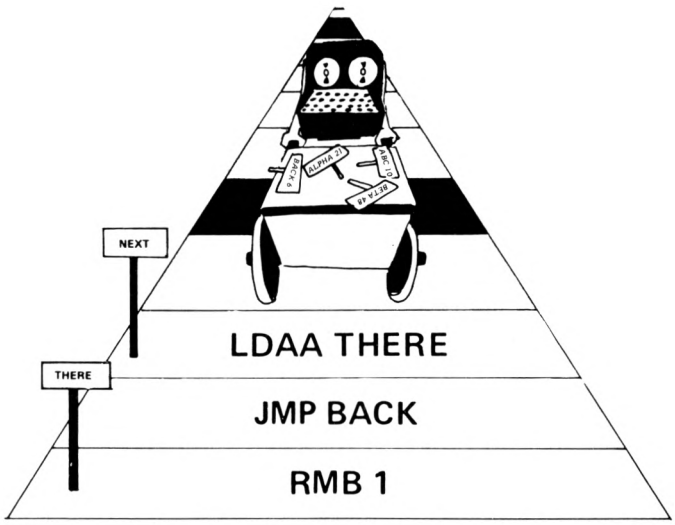
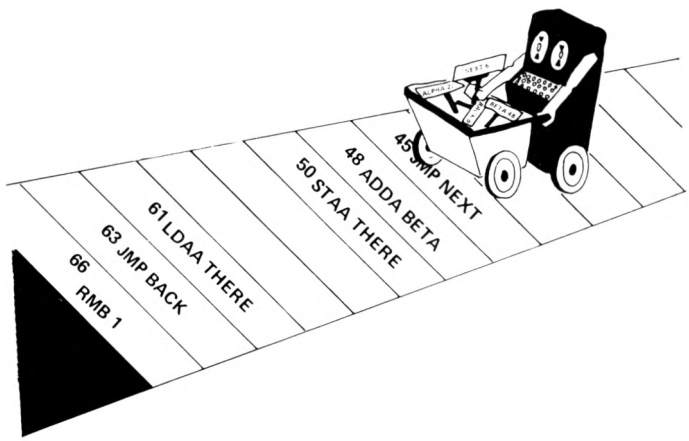


Fig. 3. Deuxième passage génération du code référençant les étiquettes.



— à construire la table des symboles et à y faire des opérations de recherche.

Si nous pouvons trouver des manières simples et efficaces de traiter ces opérations et d'éviter de les entrelacer irrémédiablement avec le reste de la logique du programme, nous arriverons à un assembleur tout à fait honnête. Bien, maintenons jetons un œil aux techniques de programmation pour l'exploration et la recherche des tables de symboles.

Techniques d'exploration

La première tâche de notre assembleur est d'analyser les caractères constituant un programme en langage d'assemblage et de trouver des choses telles que des mnémoniques d'instruction, des constantes et des noms définis par le programmeur, en notant (mais en général en ignorant) les blancs, les symboles de ponctuation et les commentaires. La première impulsion du programmeur amateur est habituellement de se jeter dans l'écriture d'une série de tests et de branchements pour traiter les différentes séquences de caractères qui peuvent apparaître sur une ligne. Cette approche conduit fréquemment à un type d'explorateur connu comme "sac de nœuds". Le scientifique de l'ordinateur, par contre, n'a rien d'autre que du mépris pour cette approche "maladroitement structurée", et préfère travailler avec des expressions régulières ou des grammaires linéaires et des automates à états finis. Nous devons prendre une solution intermédiaire, soulignant quelques techniques de programmation qui aideront à se faire la main sur la création d'un explorateur plus simple, plus petit et plus rapide.

La première technique, si vous êtes en train d'étudier votre propre langage d'assemblage, est de le rendre simple à analyser ! Une instruction de langage d'assemblage consiste habituellement en une étiquette optionnelle de l'instruction (qui représente ensuite l'adresse de l'emplacement dans laquelle l'instruction est assemblée), un mnémonique d'instruction, une zone opérande, et de la place pour les commentaires. Un exemple classique serait :

			DEBUT
			FONCTION
EVAL	**LDAA**	**BETA**	**EVALUATION**
étiquette de l'instruction	mnémonique de l'instruction	zone opérande	commentaires

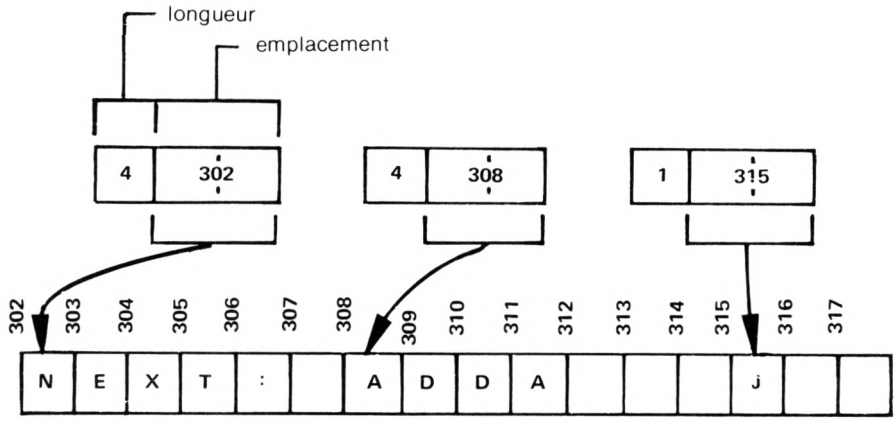
Quelques assembleurs demandent à chaque élément d'une instruction de commencer dans une colonne fixée ou à une position de caractère d'une ligne,, de telle façon que le problème d'adresse des éléments pour l'exploration soit grandement simplifié. Cependant, ceci est un peu pénible pour l'utilisateur du langage d'assemblage, mais vous allez probablement gagner du temps dans l'ensemble en mettant en place un explorateur légèrement plus complexe. Pour permettre un format plus souple, on peut prendre soit l' "approche IBM", dans laquelle une étiquette d'instruction doit commmencer en colonne 1, un mnémonique d'instruction doit être précédé par au moins un blanc et les commentaires sont séparés des opérandes par un blanc ; soit l' "approche DEC" dans laquelle les étiquettes des instructions sont suivies de deux point (:) (ou d'un autre symbole de ponctuation) et où les commentaires sont précédés d'un point virgule. L' "approche DEC" est quelquefois plus adaptée et moins propice à l'erreur pour l'utilisateur, mais est relativement plus difficile à analyser. Par exemple, on peut avoir à analyser une chaîne de caractères alphanumériques suivis de blancs, et on attendra le deux-points ou un caractère alphabétique pour pouvoir décider si la chaîne était une étiquette d'instruction ou un mnémonique d'instruction.

Quelquefois, une décision sur ce qu'il faut faire ensuite peut reposer sur le type du caractère suivant (non blanc). Si plusieurs alternatives sont possibles, on aimerait utiliser une "table des sauts", ou un tableau des adresses de branchement indexé par le code caractère, au lieu d'une séquence de comparaison de caractères. Mais l'ensemble des caractères ASCII permet 128 codes caractères différents, dont environ 45 seulement sont utilisés dans les instructions du langage d'assemblage. D'où une technique classique pour les problèmes d'exploration qui est de remplacer en premier l'ASCII par un ensemble plus pratique de codes caractère, utilisant une table de traduction de 128 caractères d'un octet. Les nouveaux codes caractère peuvent être choisis de façon à faciliter l'emploi des tables de saut en d'autres points.

Les éléments d'une instruction en langage d'assemblage (noms, mnémoniques et constantes) consistent généralement en chaînes de caractères de longueurs variables, séparées par un nombre variable de blancs. Les ordinateurs actuels, cependant, sont plus adaptés au traitement d'éléments de tailles fixées tels que les octets et le mots. Ainsi la technique la plus importante que vous pouvez utiliser pour que votre explorateur reste cohérent est d'écrire un programme "signe suivant" qui explore une chaîne alphanumérique, une constante (par exemple une chaîne de digits) ou un symbole de ponctuation chaque fois qu'il est appelé. Ce programme renverrait *un signal* suivant le type d'élément d'information ou de signe juste visualisé (disons 1 pour des chaînes alphanumériques, 2 pour des chaînes de digits, 3 pour un point, 4 pour une virgule, etc.), et *un descripteur* de taille fixée donnant l'adresse du premier caractère et le nombre de caractères dans la chaîne.

La figure 4 illustre les descripteurs pour l'étiquette, le mnémonique et l'opérande d'une instruction classique du langage d'assemblage.

Fig. 4. Les descripteurs identifient les signes d'un texte dans une ligne de caractères.



Les descripteurs de chaînes de caractères sont maniables pour pas mal d'applications. Des routines de mouvements de chaînes de caractères et des programmes de comparaisons peuvent être écrits, ce qui nécessite deux descripteurs comme arguments. Les lignes de sortie peuvent être construites à partir d'une séquence de descripteurs, et des messages d'erreur peuvent également être utilisés de cette façon. En emmagasinant les descripteurs de tailles fixées dans la table des symboles, et les chaînes de caractères elles-mêmes, dans une autre zone, vous pouvez éviter la restriction arbitraire sur la longueur des noms (de 6 à 8 caractères) rencontrée dans beaucoup d'assembleurs.

Encore plus important, l'utilisation d'un programme "signe suivant" sépare les détails d'exploration de caractères particuliers, du problème de décider comment exécuter chaque élément de l'instruction. Les programmes des tables de symboles décrits ci-dessous séparent de la même façon les détails d'identification des noms particuliers et des mnémoniques des autres problèmes de traitement. Ce sont des exemples de l'emploi de la modularité et de la structure hiérarchique pour organiser la solution d'un problème complexe.

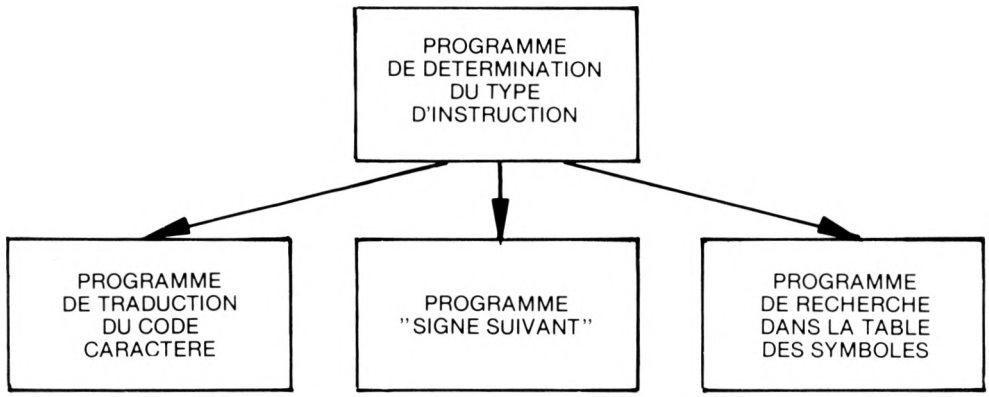
Assez de généralisations et de philosophie ; continuons avec un exemple pour voir comment tout cela fonctionne. La figure 5 montre la circulation d'information d'un programme de traduction de code caractère à un programme signe suivant, et à un programme qui détermine le type d'instruction du mnémonique de l'instruction en utilisant un sous-programme de recherche dans la table des symboles. Le langage d'assemblage de l'INTEL 8080 a été utilisé dans cet exemple. Les lettres minuscules sont traduites en majuscules, et les codes pour les chiffres (0-9) et les lettres (A = 1, B = 11, ..., Z = 35) sont choisis de façon à ce que les chaînes alphanumériques et numériques puissent être balayées en utilisant une seule comparaison pour chaque caractère. Notons l'emploi d'une table de saut "BRTAB" pour sélectionner le sous-programme de manipulation approprié pour le caractère suivant dans le programme signe suivant. Les descripteurs sont renvoyés au programme de détermination du type d'instruction, et sont passés au programme de recherche dans la table des symboles qui les utilise pour des comparaisons de caractères. Le problème de la distinction des étiquettes d'instruction suivies par un deux-points est facilement résolu à ce niveau : le signe suivant est obtenu, et son descripteur est sauvegardé ; le signe suivant est obtenu, et son code est testé ; si un deux-points a été trouvé, le descripteur sauvegardé est envoyé au programme de recherche de la table des symboles, et deux autres signes sont obtenus pour compenser avant que l'instruction ne soit traitée.

Tables de symboles

L'avantage principal que procure un assembleur au programmeur, c'est sa possibilité de donner des noms aux adresses mémoire et de faire référence à ces noms en différents points du programme. L'assembleur détermine l'adresse absolue de l'adresse mémoire, et remplit l'adresse chaque fois que le nom est référencé.

L'assembleur accomplit ceci en construisant une table de symboles lors du premier passage. Chaque élément de la table des symboles contient un nom défini par le programmeur sous la forme d'une chaîne de caractères, et l'adresse binaire correspondante. De plus, la table des symboles peut contenir d'autres noms de chaînes de caractères, tels que les mnémoniques d'instruction ou les pseudos de l'assembleur. L'élément pour le mnémonique d'une instruction devra contenir le code opération binaire correspondant, et l'élément d'une pseudo contiendra l'adresse d'un programme de traitement dans l'assembleur lui-même. Pour un ordinateur ayant plusieurs formats différents d'instructions, l'élément d'un mné-

Fig. 5. Traduction classique de caractère (8080) et programmes "signe suivant".



PROGRAMME DE TRADUCTION DE CARACTÈRES

```
        MVI     H, TABLE  H → table de contenu de page
        LXI     D, LINE   DE → début de ligne
        MVI     C, 72     C = longueur de la ligne
LOOP :  LDAX    D         prendre caractère suivant de la ligne
        MOV     L, A      L = index du code caractère
        MOV     A, M      A = élément de la table correspondant
                          à l'index
        STAX    D         remettre caractère dans ligne
        INX     D         aller au caractère suivant
        DCR     C         diminuer le nombre de caractères restants
        JNZ     LOOP      boucler pour les 72 caractères
```

PROGRAMME " SIGNE SUIVANT "

```
        LXI     H,BRTAB   H → début de la table de branchement
        LDAX    D         prendre caractère traduit dans ligne
        RLC               doublage pour indice de la table
                          de branchement
        MOV     B, A      mettre de côté 2n dans B
        LDAX    D         rajouter n pour avoir
        ADD     B         3n
        MOV     C, A      créer l'indice de 16 bits
        MVI     B, Ø      dans BC
        DAD     B         ajouter pour se brancher au début
                          de la table
        PCHL              sauter au sous-programme approprié
BRTAB : JMP     LETTER
        JMP     DIGIT
        JMP     COLON
        JMP     COMMA
        •
        •
        •
LETTER: XCHG              HL → début de la chaîne alphanumérique
        SHLD    DESCR+1   mettre adresse de départ dans descripteur
        MVI     A, 36     code max traduit pour alphanumérique
        MVI     C,Ø       initialiser compteur de caractères dans
                          la chaîne
SCAN :  INX     H         passer au caractère suivant
        INR     C         augmenter le compteur de caractères
        CMP     M         code < max pour les caractères
                          alphanumériques
        JP      SCAN      continuer à explorer
        LXI     H,DESCR   HL → partie longueur du descripteur
        MOV     M, C      mettre le nombre de caractères de la chaîne
```

monique d'instruction contiendra aussi un code de type indiquant le format propre à cette instruction, le nombre d'opérandes possibles, et l'interprétation des opérandes par des adresses ou des valeurs.

La manière la plus simple d'organiser la table des symboles serait de la représenter comme un tableau de descripteurs et de mots adresse, comme l'illustre la figure 6. Les entrées sont ajoutées séquentiellement au tableau pendant le premier passage, et une recherche séquentielle de tout le tableau est faite pour trouver les adresses des noms définis par le programmeur au cours du deuxième passage. (Tous les descripteurs provenant de la table passent à tour de rôle par un programme de comparaison de caractères, avec le descripteur comme opérande. La comparaison sera immédiatement mauvaise si les longueurs des chaînes dans les descripteurs sont inégales). Ce type d'organisation a l'avantage d'être d'une très grande simplicité, et est probablement adéquat pour une première version de votre propre assembleur. Au fur et à mesure que les programmes à assembler deviennent plus longs, l'assembleur passera de plus en plus de temps pour la recherche dans la table des symboles. On aura donc besoin d'une méthode de recherche en table plus rapide.

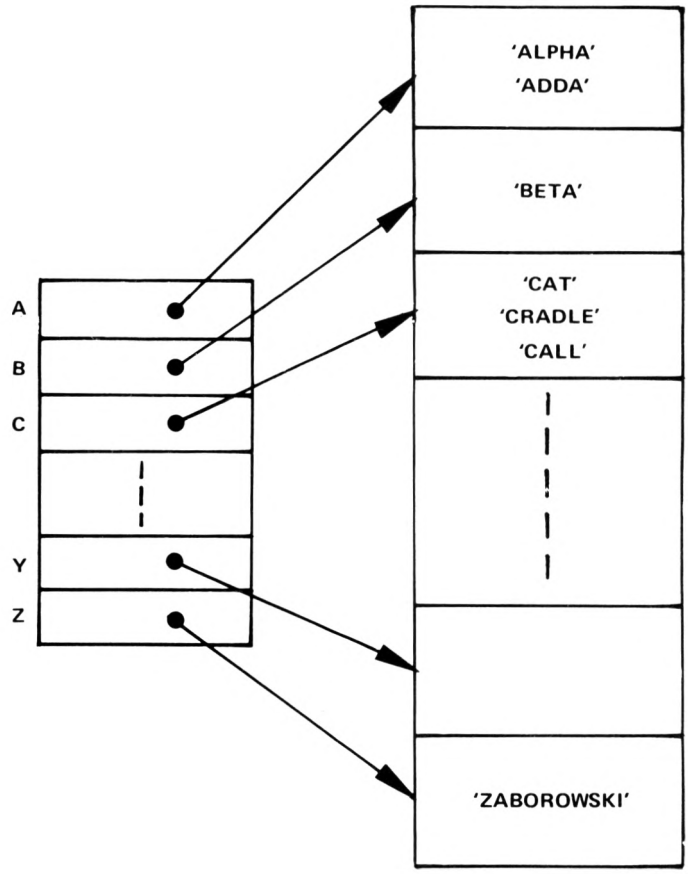
Pensez à la façon dont vous vous y prendriez pour une telle recherche, si vous étiez l'assembleur. Que faites-vous lorsque vous ouvrez un dictionnaire ou un annuaire téléphonique ? Connaissant l'ordre de l'alphabet et l'épaisseur du livre, vous regardez le premier ou les deux premiers caractères du mot, vous faites une estimation de la page, vous ouvrez le livre à cette page, et vous commencez la recherche à partir de ce point.

Laissons l'assembleur faire la même chose. Nous allons diviser la table en 26 sections, une par lettre. Nous connaissons l'adresse du début de chaque section de la table (nous pouvons faire un petit tableau des 26 adresses de début), donc pour chercher un nom, nous allons regarder le premier caractère, aller à la section appropriée et rechercher seulement dans cette section plutôt que dans toute la table.

Cette approche est décrite par la figure 7.

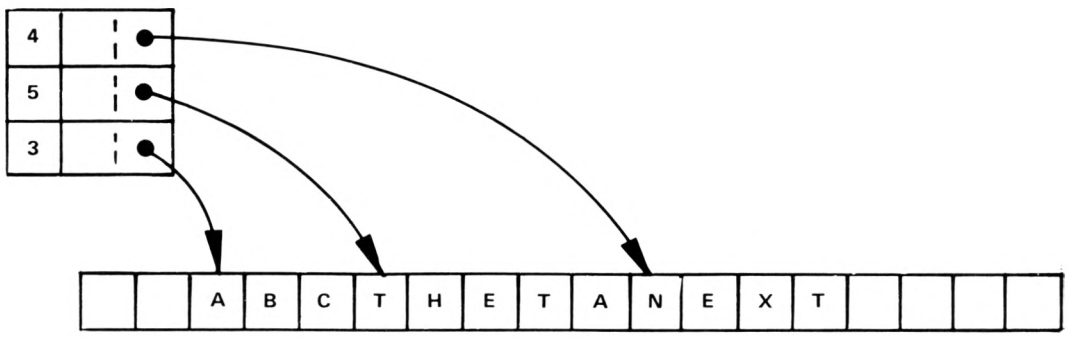
Ceci est bon pour un premier essai, mais il y a quelques inconvénients. Dans un programme appelé disons "assembleur", vous devez avoir un nombre important de noms commençant par A, tout comme

Fig. 7. Une table de symboles indexée alphabétiquement.



dans un programme appelé "éditeur" vous aurez beaucoup de noms commençant par E. D'autre part votre ami Zaborowski fera commencer tous ses noms

Fig. 6. Une table de symbole sous forme de tableau.



par Z. Est-ce que toutes les sections devraient avoir la même taille ? Si ce n'est pas le cas, comment savez-vous (au début d'un assemblage) quelles sections seront plus grandes ? Si une section devient pleine, on peut simplement ajouter les noms en trop à la section suivante de la table ; mais si un nom recherché au cours du 2e passage n'est pas trouvé dans sa section propre, on devra regarder dans la plupart des sections suivantes avant de trouver le nom. On appelle ce phénomène les "touffes". Votre ami Zaborowski est justement à même de rencontrer ce problème, et même si vous faites la section Z suffisamment large, la recherche dans la table des symboles prendra autant de temps avec la nouvelle méthode qu'avec l'ancienne.

Pouvons-nous venir à bout de ces inconvénients avec la nouvelle méthode ? Nous en sommes à un stade où une petite réflexion nous aiderait. Nous utilisons le fait que nous connaissons l'ordre de l'alphabet. Essayons de faire l'approche inverse : qu'arrive-t-il si nous utilisons un accès aléatoire aux noms en les plaçant au hasard dans les différentes sections de la table ? Au premier abord cela semble absurde, mais en y regardant de plus près nous réalisons que cela résout notre problème ! Le problème naît du fait que les gens sont pratiquement obligés de choisir un ensemble de noms qui soit en rapport avec l'ordre alphabétique ; en utilisant un ordre de choix aléatoire, nous pouvons minimiser la probabilité qu'un grand nombre de symboles soient placés dans une seule section de la table. Cette technique qui est appelée "hashing" ou "adressage aléatoire" est utilisée, pour différentes raisons, dans tous les assembleurs et les compilateurs modernes.

Donc au lieu d'utiliser le premier caractère d'un nom pour choisir la section correspondante de la table, nous allons utiliser un ensemble aléatoire de bits, ou une fonction aléatoire de la combinaison de bits du nom entier, pour choisir un point de départ dans la table. On appelle une fonction de ce type "fonction aléatoire". Aussi longtemps que les valeurs possibles de la fonction seront également distribuées parmi les rangées d'adresses des éléments de la table, le problème de groupage des noms sera minimisé.

Un exemple d'une fonction aléatoire qui donne habituellement de bons résultats est d'ajouter tous les bits d'une chaîne de caractères, en ignorant les dépassements de capacité, ou de faire un "OU exclusif" de tous les bits ensemble.

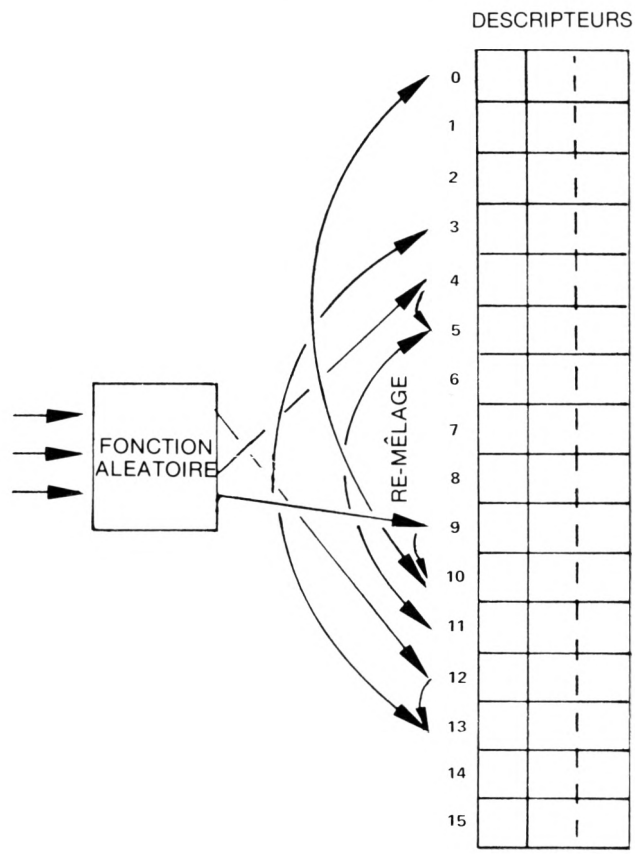
De même, dans le but de minimiser les touffes des noms qui donnent la même adresse de départ, nous pouvons "re-mêler" les noms de façon à les distribuer aléatoirement dans la table. Cette méthode est appelée "re-mêlage aléatoire". La méthode qui suit est facile à mettre en œuvre, efficace, et marche bien quand la taille de la table est une puissance de deux, 2**k (voir Morris) : supposons un nom initialement mis dans l'élément h de la table, qui est déjà occupé par un autre nom. On initialise une variable R à 1. Pour re-mêler le nom :

1 : Faire R = R*5 (décaler à gauche de deux bits, et ajouter au nombre original).
2 : Masquer tout sauf les k + 2 bits de plus faible poids de R et conserver comme nouveau R.
3 : Décaler R de deux bits à droite et l'ajouter à h pour obtenir l'élément h suivant de la table. Si l'emplacement est occupé, re-mêler le nom encore une fois.

Pour trouver un nom dans la table au cours du deuxième passage, nous allons simplement mêler et re-mêler de la même façon, en comparant cette fois chaque élément h de la table avec le nom que l'on doit trouver. Ce qui est remarquable dans cet algorithme, c'est que le nombre de comparaisons nécessaires pour trouver un élément, dépend en moyenne seulement du fait que la table est plus ou moins pleine et non pas de sa taille. Même quand la table est à 90 % pleine, seulement 2,56 comparaisons seront nécessaires en moyenne. Par contre, pour une table presque pleine de 512 éléments, la méthode de recherche séquentielle décrite plus haut prendra environ 256 comparaisons, soit à peu près 100 fois plus de temps !

Le re-mêlage aléatoire est illustré sur la figure 8. Le premier nom à mêler étant l'élément 12 de la table,

Fig. 8. Descripteurs de la table de symboles mêlée.



par exemple, il y serait enregistré, alors que le nom suivant, dont la valeur de la fonction de mêlage serait 12, serait re-mêlé pour donner l'élément 13 de la table, enfin celui-ci serait re-mêlé deux fois, et serait finalement enregistré dans l'élément 3 de la table.

Nous n'avons décrit ici qu'une seule méthode de mêlage ; plusieurs autres façons de faire sont possibles. La plus importante est appelée "mêlage avec chaînage de dépassement de capacité". Dans cette méthode tous les noms qui sont mêlés de façon à donner la même adresse de départ sont chaînés ensemble sur une liste cyclique. Cette méthode est souvent utilisée sur de gros ordinateurs ayant une affectation mémoire dynamique, et est donc moins adaptée aux micro-ordinateurs parce qu'elle nécessite une zone adresse supplémentaire pour chaque élément de la table des symboles. Les références à la fin de l'article peuvent être consultées pour plus de détails sur le mêlage.

Maintenant que vous vous êtes initiés à quelques unes des techniques fondamentales de programmation utilisées pour l'exploration et la recherche dans des tables de symboles, vous êtes pratiquement prêt pour commencer à écrire votre propre assembleur ! Vous désirerez peut-être essayer, en utilisant les techniques les plus simples vues dans les grandes lignes dans cet article : peut-être un explorateur à colonnes fixées et une table de symboles à recherche séquentielle pour une première version. Très souvent, lorsqu'on vient de véritablement écrire un programme et de le faire tourner, il arrive que la méthode la plus simplement pensée s'avère être celle qui fonctionne le mieux. Une fois que vous aurez obtenu un assembleur qui marche, vous pourrez envisager d'ajouter quelques-unes des fonctions que nous allons discuter maintenant.

Quelques fonctions supplémentaires

Jusqu'à maintenant nous nous sommes intéressés uniquement aux fonctions de base d'un assembleur : la conversion des mnémoniques et des noms définis par le programmeur en codes-op d'instruction et en adresses. Beaucoup d'autres fonctions peuvent être ajoutées à un langage d'assemblage, afin de le rendre de plus en plus pratique pour programmer. Quelques-unes des plus utiles seront considérées ici.

Définition des constantes

La plupart des langages d'assemblage ont des pseudo-opérations qui disent à l'assembleur de réserver une ou plusieurs adresses contenant des constantes. Par exemple, le langage d'assemblage du Motorola 6800 a une pseudo FCB (former un octet constant). Voici un exemple de son emploi :

```
        FCB     23,$FA
```

qui réservera deux octets contenant 0001 0111 (23 en décimal) et 1111 1010 (FA en hexadécimal, en base 16). Quelquefois, une instructions prend son opérande dans une adresse mémoire (plutôt qu'un opérande "immédiat"), mais l'opérande lui-même est en fait une constante. Au lieu d'écrire :

```
                ADDA            TROIS
                 •
                 •
                 •
TROIS           FCB             3
```

on aimerait pouvoir écrire

```
ADDA = 3
```

et avoir l'assembleur qui réserve automatiquement une adresse mémoire contenant 3, et qui assemble cette adresse dans l'instruction. Un tel opérande d'instruction est appelé un "littéral". Sur les machines où certaines instructions peuvent adresser seulement un nombre limité d'adresses mémoire, cette possibilité pourra être difficile à mettre en place.

Equivalences

Il est souvent pratique de pouvoir définir un symbole pour une constante, ou avec la même valeur qu'un autre symbole. Par exemple, une constante représentant la taille d'un tableau pourra être utilisée en plusieurs points du programme. En utilisant un symbole à la place de la constante d'un bout à l'autre du programme, et en définissant la valeur de la constante du symbole au début du programme, nous pouvons rendre plus facile le changement de la taille du tableau quand on réalisera une nouvelle version : seul le symbole a besoin d'être redéfini, et sa nouvelle valeur sera substituée aux endroits appropriés par le processus normal d'assemblage. (On appelle cela la "paramétrisation du programme"). Cette possibilité n'est pas trop difficile à mettre en place, et la plupart des assembleurs ont une pseudo-op de la forme

```
DIMT    EQU     25
```

qui nous permet d'écrire

```
                LDAA            = DIMT
                 •
                 •
                 •
TABLE           RMB             DIMT
```

ou, en général on utilisera le symbole DIMT partout où la constante 25 devrait apparaître.

Evaluation d'expressions

En plus de la définition des constantes et des symboles à valeur constante dans un programme, il est souvent utile de pouvoir combiner de tels éléments dans des expressions arithmétiques dont les valeurs

peuvent être calculées au moment de l'assemblage, et de pouvoir utiliser ces valeurs à la place d'autres constantes. Par exemple le même programme avec un paramètre DIMT pour la taille d'un tableau pourrait comprendre des instructions telles que :

```
        ADDA        #DIMT—1
         •
         •
         •
SPACE   EQU         3*DIMT+1
```

qui spécifieraient (pour DIMT = 25) que 24 sera ajouté à l'accumulateur A, et que SPACE aura la valeur constante 76 partout où il apparaîtra dans le programme.

Il est remarquablement facile d'évaluer des expressions de cette sorte, en tenant compte des parenthèses et de la priorité normale des opérations arithmétiques. (Vous pouvez consulter Mealy ou Gries, voir les références).

Assemblage conditionnel

Nous avons vu comment un programme pouvait être paramétré par l'utilisation de symboles équivalents et d'expressions arithmétiques. Parfois un programme peut être paramétré d'une autre façon : des sections entières du programme peuvent être incluses ou omises, et ceci dépendant de la valeur de certains paramètres. Par exemple, si la valeur maximale d'une certaine variable est inférieure à 256, elle peut être enregistrée dans un seul octet sur la plupart des machines ; mais si la valeur maximale est supérieure ou égale à 256, il faudra deux octets ou un mot. On pourrait ainsi espérer écrire quelque chose comme :

```
        .SI     MAXVAL   LT  256
TABLE   RMB     DIMT
        .FIN
        .SI     MAXVAL   GE  256
TABLE   RMB     2*DIMT
        .FIN
```

avec comme objectif que si une pseudo EQU a défini plus haut MAXVAL disons à 200, la première instruction RMB sera assemblée, et que si MAXVAL a été défini disons à 400 la seconde RMB sera assemblée.

Cette possibilité n'est pas trop difficile à mettre en œuvre, et elle est extrêmement utile. L'assembleur doit simplement reconnaître les pseudos SI et FIN, évaluer les relations et sauter le texte à chacun des deux passages si la relation est fausse. Il est facile d'imaginer (mais un peu plus difficile à réaliser) des extensions à cette possibilité, telles que l'assemblage répétitif de certains segments de programme.

Macros et translation

Les assembleurs les plus sophistiqués sont comparables à des compilateurs, en complexité, en taille et en souplesse. Certains assembleurs permettent d'utiliser une macro permettant au programmeur de définir de nouveaux mnémoniques d'instruction, qui seront remplacés par des séquences paramétrées d'instructions de langage d'assemblage partout où ils apparaissent dans le programme. Quand on combine ceci avec les possibilités d'assemblage conditionnel, cette possibilité de *macro* constitue un outil puissant pour étendre un langage d'assemblage et l'adapter à une application particulière.

Nous avons seulement parlé des assembleurs absolus : nous commençons par supposer que le programme à assembler commence à l'adresse 0 (ou toute autre adresse fixée). Lorsque le programme va être chargé en mémoire, avec d'autres programmes assemblés auparavant, nous ne savons pas si les autres programmes sont importants et dans quel ordre ils vont être chargés. Dans ce cas, il est nécessaire de mettre l'information de *translation* avec le programme assemblé, ce qui revient à dire : «si vous chargez ce programme à l'adresse *m*, vous devez ajouter le nombre *m* aux octets ou aux mots suivants pour que les adresses deviennent correctes». Cette information de translation est traitée par un chargeur, qui est responsable du chargement de tous les programmes translatés en mémoire.

Ces deux sujets sont intéressants et très importants, plusieurs pages seraient nécessaires pour leur faire honneur et cet article est déjà assez long ! Aussi allons-nous nous contenter des sujets déjà traités. Cependant, vous avez probablement décidé soit qu'écrire un assembleur est trop de travail, et vous avez arrêté la lecture de cet article, soit que l'idée est très intéressante, et vous attendez le dernier mot avec impatience. Aussi le voilà : maintenant que vous savez comment écrire un assembleur, pourquoi ne pas vous jeter à l'eau et faire un essai ? Vous n'avez rien à perdre, sauf votre innocence sur les complexités du logiciel système, et peut-être un peu de votre temps.

Bonne chance !

Références

Barron, D.W. *Assemblers and Loaders.* American Elsevier (Computer Monograph Series, n° 6), New York, 1969. Traduit en français chez Dunod *(Assembleurs et chargeurs).*

Le texte le plus complet, facile à trouver sur l'étude des assembleurs et des chargeurs ; décrit aussi les assembleurs à un passage et les méta-assembleurs.

Gries, David. *Compiler Construction for Digital Computers.* Wiley, New York, 1971.

Un texte hautement recommandé sur l'étude des compilateurs : il couvre à la fois l'aspect théorique et l'aspect pratique du problème. Il comprend une bonne discussion sur le mêlage et sur d'autres méthodes plus sophistiquées d'exploration.

Mealy, George. "A Generalized Assembly System (Experts)", in Saul Rosen (ed.), *Programming Systems and Languages.* McGrawHill, New York, 1967.

Un article classique par un des pionniers des traducteurs de langage et des systèmes d'exploitation. Il présente l'idée des descripteurs pour des chaînes de caractères, ainsi que beaucoup d'autres innovations.

Morries, R. "Scatter Storage Techniques", in *Communications of the ACM 11 :* 1 (January 1968), pp 38-44.

Un des meilleurs aperçus généraux des techniques de mêlage ; comprend une bonne et brève description du mêlage avec dépassement de capacité.

INDEX

INDEX NUMERIQUE

1 bit, 301
4 bits, 61
8 bits, 61
12 bits, 64
16 bits, 64, 287
32 bits, 300
38, 281, 283
74, 141,167
74LS, 164
360, 57, 93, 96, 117
360/30, 56, 115, 266, 267, 279, 306
360/40, 299, 302, 306
360/50, 299, 306
370, 281
370/168, 255
470, 281, 283
650, 121
704, 281
990, 288
1100/60, 284
1400, 117
1401, 115, 117, 118, 266, 267, 268, 288
1802, 290
2125, 291
2147 H, 291, 293
2200, 277
2650, 58, 97
2901 A, 290
2920, 291
3000, 61, 277, 290
3870, 290
4004, 53,54
4013 A, 64
4040, 56
4300, 285
5100, 277
6100, 64
6502 C, 290
6800, 57, 58, 72, 290, 301, 312, 319
6802, 290
6809, 301
7000, 117
7090, 134
7300, 61
7400, 141
7407, 228
7442, 167
8008, 52, 55, 56, 67, 68, 70
8021/48/39, 290
8080, 56, 72, 73, 289, 290, 315
8085, 289
8086, 293

8088, 302
8094, 167
8212, 149, 167, 218
8224, 129
8228, 149, 156
8255, 2e p. av. propos, 220
8704, 149
8708, 2708, 149, 211
9440, 293
9445, 293
9830, 277
10800, 284
68000, 300
93481, 293

INDEX ALPHABETIQUE

Accès aux fichiers, 71
Accès à la mémoire, 27, 28
Accumulateur, 50, 55, 86, 312
Adressage aléatoire, 318
Adressage direct, 96
Adressage indirect 69
Adressage (mémoire, périphérique), 165, 214, 218, 296, 299
Adressage relatif, 268
Adresse, 52, 70
Adresse de décalage, 297
Adresse d'entrée/sortie, 214
Affichage graphique, 264
Alcool isopropylique, 194
Alcyane, 73, 78, 97, 112, 115, 214, 261, 262, 271, 303, 306, 307
Aléatoire, 318
Algèbre de Boole, 21
Algol, 289
Algorithme de mêlage, 318
Alimentation, 145, 153, 171, 305, 307
Alpha microsystems, 289
Alpha (particules), 303, 304, 305
Alternative, 283
Alu, 50
Alumine, 40
Aluminium, 38
Aluminium-cuivre, 281
AM-100, 289
AMD, 290, 299
AMDAHL, 281, 283
American microsystems inc., 61, 291
AMI, 61, 291
Analyse syntaxique, 277
Analyseur, 278
Analyseur syntaxique (explorateur), 314
And, 20, 89
Antémémoire, 295
Apollo, 134
Appel de ligne, 278
Approche descendante, 109
Architecture nodale, 267, 271, 272
Architectures (harvard/princeton), 54, 74
Aristote, 18
Arithmétique, 83
Armonk, 281
Arrondis, 276
Arsenic, 11, 12
Arséniure de gallium, 286
ASC II, 56, 169, 232, 275, 312, 314
Assemblage, 140, 311, 312, 313, 316
Assemblage conditionnel, 320
Assembleur, 83, 90, 98, 311, 312, 313, 316
Astérisque, 109

Asynchrone, 181, 231, 232, 234
Atomes, 9
Auto-alignement, 44
Avalanche, 13
Azerty, 123

Babbage, 18
Bach, 2e p. préface
Balayage (données de), 283
Balayage (scanning), 64, 66, 168
Bardeen, 9, 14
Barre, 19
Barron (D.W.), 320
Bascule, 30, 31, 283, 284
Bascule R/S, C, D, J-K, 31 à 34, 218, 283, 301
Bascule à verrouillage, 71
Base, 14, 281
Basic, 196, 268, 274, 276
Baud, 181, 233
Baudot, 181
Beam lead, 39
Bi-directionnel, 165, 220
Bifréquence, 181
Binaire, 275
Bipolaire, 41, 292, 293
Bipolar, 41
Bistable, 31
Bistable latch, 71
Bit, 18
Bit start, stop, 184
BIU, 296
Bobine mobile, 269
Boîtier calculateur, 53
Boîtiers, 45, 255, 258, 307
Book (livre gabarit), 282
Boole, 18
Booleen, 21
Boorman (John), 286
Bottom-up, 109
Boucle de courant, 238
Bouhot, 7, 18, 116, 117
Branchement, 99, 297
Brattain, 9, 14
Break, 278
Bruit électrique, 260, 262, 305
Buffer (registre tampon), 51, 73, 179, 216
Bufferiser, 218
Bull, 268, 271, 307
Bureaux d'étude, 2
Burroughs, 118
Bus, bus externe, 51, 52, 70, 73, 144, 165, 214, 215, 217, 302
Bus commandes, 215
Bus information, 215
Bus interne, 51
Bus multiplexé, 52

Câblé, 57
Cache (mémoire), 285
CAD, 40
Caisson d'isolation, 37, 39
Call, 79, 102
Canal (P, N), 42, 43, 291
Canaux, 306, 307
CAD, 40, 282
Capacité parasite, 134
Carroll (lewis), 18
Carry, 86
Carte, 129
Cartes perforées (capacités), 306
Cartes (taille des), 262
Cartouche amovible, 269
Catalogue, 208
Cathode, 15, 143
Cathodique (tube), 40
Cavalier, 148
CCITT, 231
CCITT V24, 234, 267
Céramique, 254, 303
Chaînage de dépassement de capacité, 319
Chaînes de caractères, 113, 114, 267, 274, 298, 314
Charge d'espace, 12, 15
Chip, 35, 53
Chrome-cuivre-or, 281
Chronogramme, 77, 262
Circuit imprimé, 129, 132, 258, 262
Circuits intégrés bipolaires, 34 et ses, 41, 254
Circuits intégrés monolithiques, 35
Circuits intégrés mos, 41, 43
Circuit logique, 18
Claviers, 123, 124, 168, 277
C/MOS, 47, 290, 301
CMP-8, 61
CO, 58, 61, 69, 70, 74
Cobol, 275, 278
Code de couleur, 142
Codes instructions, 102
Code à 11 moments, 233
Code pièces détachées, 271
Coffret, 146
Collecteur, 14, 281
Communication, 213
Communications register unit, 288
Commutation, 15, 285
Commutation téléphonique, 119
Comparateur de signes, 21
Compatibilité, 47
Compilateur, 279, 289, 313, 320, 321
Complementary MOS, 47
Complément à deux, 83
Complément à dix, 83
Complément à un, 83
Compteur ordinal, 50
Compteurs à propagation, 34
Computer aided design, 40
Computers and people, 126
Conception, 127, 253, 255, 259, 262, 280, 303, 304
Conception assistée par ordinateur, 40
Condensateurs, 142
Condensation, 275
Condition, 86
Conduction intrinsèque, 11
Conduction N, P, 12
Connecteur V24, 235
Connery (sean), 286
Connexions poutrelles, 39
Continue, 197
Contraction de dimensions, 291
Control Data, 257
Contrôle de processus, 64, 263
Control read-only memory, 59

Contrôle en saisie, 275
Contrôleur de disques, 268, 272
Conversationnel, 268, 272
Convesion, 275
Convivialité, 125
Corindon, 40
Cosmac, 64
Cottin (Georges), 1re p. préface
Couche d'arrêt, 12
Couches de logique, 46
Coût, 128
Cray-1, 257, 258
Cray (Seymour), 257, 258
Cristal, 11, 45
Crom, 59
Cru, 288
Curseur, 277
Cycle instruction de base, 70, 71
Cycle machine, 70, 71, 75, 76
Cynthia, 269, 271, 307

D, 32
Data division, 278
Data general, 134, 287
Data in/out, 52, 235
Datanet 30, 119
Datatron 205, 118
Deadly embrace (étreinte mortelle), 299
Dec-20, 306
Décalage, 33
Décimale (correction), 74, 87
Décodeur binaire-décimal, 23
Découplage, 47
Délai de propagation, 46, 282, 291
Democrite, 9, 17
De Morgan, 18
Densité de logique, 253
Densité en portes, 254
Depletion transistor, 45
Descripteur, 314, 315, 320
Design, 128, 129
Dialogue, 121
Dialogue processeur/périphérique, 224
Dielectrique de grille, 43
Diffusion, 37, 282
Diffusion self-aligned MOS, 291
Digital equipment corp., 50, 289, 314
Digitalisé, 41
Dijkstra (Edsger W.), 299
DIL, 138, 140, 141
DIM, 198
Diode, 9, 172
Diode adonction, 12
Diode électro-luminescente, 13, 143
Diode laser, 13
Diode shunt shottky, 282
DIP, 45, 138, 141, 146
Direct memory access, 74
Dispatcher, 270
Disques rigides, 267, 268
Disques souples, 202
DMA, 74, 301
D-MOS, 291
Données, 52, 297, 299
Donneur, 11
Dopé, 11, 35, 37
Drain, 42, 43, 292
Drapeaux (flags), 102
Driver (soft), 2e p. av. propos, 168, 215
DSA-MOS, 291
DTL, 46
Dunod, 320
Dvorak, 124
Dynamique, 57, 134, 289, 293, 301, 303
Dynamiquement relogeable, 296
D-120, 271

Ebcdic, 56
Ecl, 46, 284
Eclipse, 134
Economie d'échelle, 7
Ecriture/lecture (K7), 184
Ecriture mémoire, 52
Ecriture périphérique, 52
Editions radio, 127
Efcis, 300
Effet de poinçonnage, 307
Effet de seconde grille, 307
Effet tunnel, 285
Electricité statique, 135, 138, 305
Electrique, 218, 228
Electronic arrays, 61
Electrons, 10
Electrons de conduction, 10
Electrons de valence, 10
Elément, 10
Emetteur, 14, 232, 237, 281
Enregistrement (mode d'), 181
Enseignement, 2
Entrance (fan-in), 286
Entrée/sortie, 112, 211
Entrée/sortie 8 bits, 228
Entrée/sortie parallèle, 213, 220
Entrée/sortie série, 213, 231, 238, 243, 244
Entreprises (services d'), 4
Epitaxie, 35, 281, 291
Equilibrage d'alimentation, 175
Equivalences, 319
Erreurs mémoire, 303
Erreurs dans un ordinateur, 305, 306
Erreurs (récupération d'), 275
ET, 19, 20
Etain-plomb, 281
Etat logique, 18, 284
Etats machine, 70, 71, 75, 77, 80
Etat travail/repos, 237
Etiquettes, 313
Etreinte mortelle (deadly embrace), 299
EU, 295, 296
Evaluation d'expressions, 319, 320
Examen mémoire, 159
Exécuteur, 278
Exécution en pas à pas, 78, 79, 159, 278
Exécution unit, 295, 296
Explorateur (analyseur syntaxique), 314
Exterminateur, 286
Exxon, 289

Facture, 272

Faggin (federico), 56, 81, 289
Fairchild, 53, 61, 290, 293
Familles logiques, 46
Fan-in (entrance), 286
Fan-out (sortance), 286
Farmer (philip-josé), 25
Feeney, 56
Fenêtre de lecture, 232
Fer, 9, 10
Fer à souder, 139
Fet, 41, 43
Fetch (cycles), 80
F-8, 61
Fiabilité, 205, 261, 305, 306, 307
Fiche de stock, 125
Fichiers, 206, 207, 272
Fichier index, 271
Field effect transistor, 41
Flag, 102
Flanc (montant, descendant), 225, 226, 231
Flat-pack, 66
Flip-flop, 30, 31, 65
Floppy, 204
Flottant (bus), 75
Fluctuations de l'alimentation, 305
Fluorhydrique (acide), 36
Fond de panier, 144, 155, 182
Format, 81, 277, 278
Format d'écran, 274
Formattage, 206
Fortune, 117
France-Lanord, 7, 116
Fuite (courant de), 12
Full duplex, 238

Galène, 9, 14
Gallium, 12, 286
Gamma 30, 288
Gate, 43
Gate (grille), 43, 291, 307
Gate (latch), 77
Gater, 77
Geller, 64
General electric, 13, 119
Génération de caractères, 179
Générations de microprocesseurs, 259
Générations d'ordinateurs, 5, 253
Géométrie (horizontale, verticale), 37
Germanium, 10
Gernet, 146
Gestion, 114, 116, 263, 267
Gestion clavier, 66
Gheewala (tyshar), 285
Gibi, 102, 103
Gibson mix, 114, 116, 267
Graphique (affichage), 264
Grenat d'yttrium, 286
Gries (david), 320
Grille, 15
Grille, 42, 43, 291, 307
Grille d'écran, 274
Grille silicium, 44, 45, 291
Grosch, 7, 116, 117

Hand shake, 224, 228
Hardware, 161
Harmonique, 48
Harvard, 54
Hashing, 318
Heathkit, 64
HEC, 135
Helium liquide, 286
Helium (noyau d'), 303
Hewlett-Packard, 61, 261
Hexa, 82, 98, 99, 100, 104, 105, 169
Hiérarchie d'ordinateurs, 279
Hiérarchique (niveau), 271
H-MOS, 291, 293
Homme, 125
Horloge, 21, 48, 51, 53, 60, 64, 66, 76, 279, 283, 290
HP-35, 61
HP-65, 61
Hybride (technologie), 254

I2L, 47
I3L, 291, 292, 293
IBM, 2e p. av. propos, 117, 202, 254, 285, 306, 314
IC 2108, 64
IEEE, 120
IF error, 277
Imp-16, 59, 60, 72
Implantation, 40, 129
Imprimante, 245, 249, 250
In/out, 277
Incrémenteur d'adresse, 74
Index (fichier), 271, 317
Industrie informatique, 256
Informaticiens, 124, 126, 280
Informatique répartie, 279
Informatique (services), 2
Input, 277
Input-output control system, 117, 118
INS 8900, 287
Instructions à deux adresses, 288
INT, 160
Intel, 52, 53, 56, 61, 67, 73, 289, 290, 291, 293, 302
Interactive (mise au point), 277
Interconnexion, 281
Interface logique, 211
Interface parallèle, 213, 245, 247, 248
Interface série, 213
Interpréteur, 278, 289
Interrogations, 272
Interrupteurs, 141
Interruptions, 69, 72, 74, 78, 110, 118, 119
Interruptions internes, 118
Intersil, 64
IOCS, 117, 118
Ioniques (semiconducteurs), 14
Ions (faisceau d'), 291
Ions sodium, 44
Iridium-plomb-or, 286
Isolation (caissons d'), 37, 39

Jeux, 4
J-K, 33, 34
Jonction, 13, 285
Jonction Josephson, 285

Josephson (Brian), 285

Key-in, 277
KIM-1, 290
Kit, 2e p. av. propos, 128
Knuth (Donald), 121, 271

Langage machine, 93, 311, 312, 313
Langages de programmation, 274
Large scale integration, 49, 256
Laser, 286
Latch, 59, 68, 71, 283
Lecarpentier, 136
Lecture mémoire, 52
Lecture périphérique, 52
LED (light-emitting diodes), 13, 139, 204
Liaison périphérique, 212
Lignes de commande, 52, 215, 224
Ligne téléphonique, 234, 237
Ligne de transmission, 232, 237
Lilen (henri), 127
Lilien feld, 43
Limite ultime, 309
Listing de programme, 278
Lithogravure à projection, 291
Littrel, 7
Livre (gabarit), 282
LM, 153, 162
LM 323K, 172
LM 340K, 162
Load, 196, 197
Lock, 298
Logiciel système, 2 e p. av. propos, 320
Logique, 17, 18 , 50
Logique combinatoire, 30
Logique à injection de courant, 285
Logique à injection intégrée, 292, 293
Logique négative, 24, 47
Logique saturée ou non, 46
Logique séquentielle, 30
Logique trois états, 168
Loi de Bouhot, 7
Look-ahead, 295
Low power schottky, 168
LS, 168
LSI, 49, 256
Lussato, 7, 116

M1, 160
M80, 281
Macro-micro, 122
Macros, 320
Magnétocassette, 181
Magnuson systems, 281
Maintenance, 274
Mannesmann, 245, 249, 250
Masque (circuits imprimés), 132
Masque (circuits intégrés), 35, 40, 74, 291
Masques d'extraction, 89
Masque projeté, 291
Masse, 48
Masse électrique, 228, 235, 246, 251
Masse mécanique, 235
Master slice, 281
Mat, 258
Matrice de décodage, 23
Mazor, 74
MC 14500B, 301
MC 68000, 300
Mc Graw-hill, 320
MCP-1600, 289
MCS-4, 53, 54, 56
MCS-8, 56, 68
Mealy (george), 320
Medium scale integration, 49
Mega-mini, 96
Mêlage, 318, 320
Mémoires, 6, 145, 152
Mémoires dynamiques, 134, 293, 301, 303
Mémoire morte de commande, 59
Mémoires statiques, 134, 301, 303
Memory management unit, 299
Mercure, 286
Messages d'erreurs, 277
Méthode d'accès, 271
Michels, 96
Micral, 52
Micro-diagnostic, 60
Microdisquettes, 202, 212
Micro-informatique, 115, 124
Micro-instruction, 59
Microflame, 293
Micromodule, 281
Micronova, 288
Micro-ordinateur, 116, 127, 259, 306
Microplaquette, 1, 35, 49, 254
Microprocesseurs en tranches, 57, 59, 60, 61
Microprocesseur, 1, 2, 49, 53, 68, 258, 259, 287, 299
Microprogrammation polyphase, 60
Microprogrammes, 57, 306
Minis, 287, 299, 300, 306
Mise sous tension, 156
MMU, 299
MN 601, 288
Mnémoniques, 311
Modem, 231, 234, 237, 279
Modes d'adressage, 96, 97
Modes (clefs, programme), 157, 158
Modes (entrée/sortie), 220
Modularité, 263, 279
Molybdène, 282
Moment (temps élémentaire), 233
Moniteur, 215
Monocristal, 35
Monolithique, 35
Monolithic memories, 64
Monomono, 120, 121
Monoprogrammation, 118
Monotâche, 118
Morris, 318
MOS, 6, 41, 43, 47, 291
MOS (symboles des transistors), 45
MOS technology, 290
Mostek, 61, 291, 295
Mot de commande, 222
Mot d'état, 57, 77, 79, 226
Motif (pattern), 284
Motorola, 57, 72, 73, 284, 290, 291, 294, 300, 301, 302, 312, 319

Move, 197
MSI, 49
Multiprocesseur, 298
Multiprogrammation, 118, 268
Musset, 18
Mylar, 129

N, 14
Nand, 25, 256
National semiconductor, 59, 60, 281, 287
NEC, 295
Negation, 18, 19
Nippon electric company, 295
Nitrure de silicium, 281
Nodal (système), 267, 271, 272
Non, 18, 19
Non-A, 18
Nor, 23, 25
Not, 18
Notre dame, 1re p. préface
Nova, 287, 288, 293
Noyau, 10
Noyce (robert N.), 309
Numéro de ligne, 196, 197

Octal, 82, 98, 99, 100, 104, 105
Opérande immédiat, 93
Opérandes, 297, 312
Operating system, 119, 300
Or (logique), 19, 90
Or (métal), 132
Ordinateurs géants, 257, 258
Ordinateurs moyens, 256, 280
Organisation, 280
Ou, 19, 90
Ou exclusif, 19, 67, 89, 194, 318
Ou inclusif, 19, 90
Oux, 19, 67, 89, 194
Overhead, 119
Overlay, 268
Oxyde, 42, 43, 291

P, 14
Pace, 287
Paires, 11
Pannes intermittentes, 261
Parallel processing system 4, 65
Paramétrisation de programme, 319
Parasitic engineering, 261
Paris, 1re p. préface
Parité, 86
Parité (bit de), 134
Pas à pas, 78, 79, 159, 278
Pascal, 289
Pascal microengine, 284
PC (co), 77, 100, 159
P-code, 289
PDP-8, 93
PDP-11, 72, 289
PDP-11/45, 300
Peau (effet de), 48

Pelissolo (jena-claude), 1re p. av. propos, 309
Périphérique, 113, 211
Périphériques (système oú non), 236, 237
Petit système, 165
Pfann, 16
Photo diode, 13
Photons, 13
Photoresist, 36
Photorésistance, 64
Photo-transistor, 13, 204
Pia, 72
Pia, 301
Picosecondes, 285, 286
Pile, 50, 67, 74, 76, 81, 301
Pinocchio, 2e p. préface
Pintade, 302
Pip, 57, 58, 97
Piste (cartes), 129
Piste (disques souples), 204, 205
PK, 64
Planar, 35
Plaque, 15
Plaquettes, 35
Plessey, 119
PLM, 72
Plomb-iridium-or, 282
PME/PMI, 4, 268
Poignée de main, 224
Pointeurs, 271
Pointeur de pile, 81
Polarisation inverse, 12
Pomme de terre, 14
Ponson du terrail, 1re p. av. propos
Pont résistif, 228, 231
Pop, 91
Port, 185, 282
Portabilité, 289
Porte, 18, 74, 256, 285
Porte-puce, 281
Portes logiques (représentation), 21
Porteurs, 11
Porteurs majoritaires, 12
Porteurs minoritaires, 12
Porteuse (détection de), 235, 237
Positionneur à bobine mobile, 269
Post, 18
Postes de traitement, 271
PPS-4, 52, 53, 54, 56, 65, 66, 72
PPS-8, 61
PPS-25, 53, 56
Précision de calcul, 276
Pré-série, 132
Princeton, 54
Processeur, 50, 71, 145, 151, 214, 256, 285
Processeur de fichier, 271, 273
Produit logique, 20
Program counter, 77
Pmrogrammable read-only memory, 67
Programmation structurée, 108, 109, 283
Programmation, 113, 114, 274
Programme (paramétrisation de), 319
Projection de masque, 291
Prom, 67
Pseudo-instructions, 100, 312, 315
Psychokinésie appliquée, 64
PSW, 57, 91
PTT, 234

Puce, 35, 259, 281, 300, 309
Pull-up (résistances de), 47
Pupitre, 143, 157

Quad in line package, 307
Quartz, 36, 66, 179, 255
Queue (du 8086), 295, 296
Quip, 307

R2E, 52
Radiateurs, 153
Radioactivité, 303
Raffinage par fusion de zone, 16
Rafraîchissement, 134, 289, 303
Ralu, 60
Ram, 53, 54, 67
Rapidité, 114, 115
Rayons alpha, 303, 304, 305
Rayons cosmiques, 303
Raytheon, 64
Raz, 158, 160
RCA, 64, 290
Rad Buffer, 277
Ready, 76, 78, 290
Récepteur, 232
Recherche instruction, 80
Record, 198
Récupération d'erreurs, 275
Référence avant, 312
Registres, 33, 50
Registre adresse instruction, 50
Registre de commande, 220
Registres généraux, 50, 93, 295
Registre programme, 50
Registre tampon (buffer), 51
Régulateurs, 153
Réinitialiser, 30
Relogement (translation), 295, 296, 320
Remise à zéro (raz), 158
Rendement, 41
Répétition de touche, 278
Répétitive, 283
Report, 86
Réseau nodal, 272
Reset, 30, 74
Résistances, 142
Retour, 106
Retour (de), 50
Rockwell, 52, 53, 61, 67
Rolls-Royce, 206
Rom, 53, 54, 67
Rondelle, 35
Rosen (saul), 320
Rotation, 90, 91
Routine, 215
RP-16, 64
R/S, 31
RS 232C, 231
RTL, 46
Rubylith, 40
Run, 197
Rupture de séquence, 99
Rutherford, 9

Saut, 99
Sca, 284
Scaled poly 5, 291
Schockley, 14, 43
Scom, 125
Scratch pad, 50, 54
Scrutation clavier, 169
Search, 275
Secteur, 204
Sécurité, 273
Segmentation, 268
Segments, 296
Seiko, 66
Self testing and repair, 284
Sémaphore, 298
Semiconducteurs, 10
Séquence, 202
Servo-tracking, 269
Servo-traquage, 269
Set, 31
Seuil (tension de), 43
SGS-Ates, 299
Shannon, 18
Shift, 33, 283
Shottky, 43, 168, 281, 282
Shugart, 204
Sicob, 129, 135
Siemens, 295
Signal, 18
Signaux d'état, 71, 74, 75
Signe, 83, 84
Signetics, 57, 58, 97
Silice, 36, 38
Silicium, 10, 11, 16
Silicium sur saphir, 40
Silicon on sapphire, 40
Silicon systems, 291
Simplicité, 120
SLT, SLD, 254
Small scale integration, 49, 254, 258
Socrate, 17
Sodium (ions), 44
Software, 274
Sortance (fan-out), 286
SOS, 39, 40
Soudure, 139, 140, 183
Source, 42, 43, 292
Source, 278
Sous-ensembles, 273
Sous-processeur, 285
Sous-programme, 50, 67, 69, 102, 108, 109, 277, 313
Sperry rand, 64
Spool, 277
SSCI, 125
SSI, 49, 254, 258
Stack pointer, 81
Stanford, 121
Star 284
Start bit, 233
Statique, 57, 134, 301, 303
Stock, 268
Stop bit, 233
Strap, 148
Strobe time, 65
Strobe, strober, 78
Structurée, 109, 283
Structure fonctionnelle, 217, 221

Substrat, 36, 281
Superordinateur, 285
Superviseur, 118
Supports, 260, 261
Supraconducteur, 285, 286
Switch, 141
Switching time, 46
Symboles, 313, 315
Symbolisme des instructions, 85
Synchro, 179
Synchrone (liaison), 233, 234
Synchronisation, 28, 51
System 250, 119
System control adapter, 284
Système, 215
Système/38, 281, 283
Systèmes d'exploitation, 119, 300

Table mêlée, 318
Table des symboles, 313, 314, 315
Table de vérité, 19
Tableau de bord, 280
Tableau comparatif, 62
Taille des cartes, 262
Tantale, 142
Techniques d'exploration, 314
Technologies, 253
Technologie hybride, 254
Télé-diagnostic, 285
Télétype 33/35, 238, 239
Télévision, 176
Temps élémentaire de bit, 232
Temps de propagation, 304
Terminal, 235
Tests, 38
Tests (de matériel), 161, 162, 283, 284
Test mémoire, 163, 164
Texas instruments, 53, 64, 288, 292, 302
Thermocompression, 39
Thomson-efcis, 300
Thorium, 303
Tiers exclu, 18
Timing unit, 51
TLCS-12, 64
TMS-0117, 53
TMS-0119, 64
TMS-1000, 290
Tolérances, 142
Top-down, 109
Toshiba, 64
Touffes, 318
Tracking (servo), 269
Traduction de caractères, 316
Traitement de texte, 264
Tranches, 35, 281
Transaction, 272, 296
Transfert à l'adresse, 159
Transfert d'information, 212, 228
Transistor, 9, 17
Transistor à appauvrissement, 45
Transistor drift, 16
Transistor à effet de champ, 41
Transistor à enrichissement, 45
Transistor à injection, 41
Transistor à jonction, 14
Transistor mesa, 16, 17
Transistor PNP/NPN, 293
Transistor unipolaire, 41
Transition négative, 232
Transitron, 64
Translation (relogement), 295, 296, 320
Transport d'information, 212, 291
Trappes, 118
Traquage (servo), 269
Tricot, 4, 127
Triode, 15
Trois états, 168
Trou, 11, 303
Trou métallisé, 129
TTI, 8, 41, 46, 138, 168, 255, 281
TTL Schottky, 47, 149, 281, 282, 295
Tube à vide, 15
Tunnel (effet), 285
Turing, 3, 18
Two Pi, 281

Ultra-violet, 67
Unité "A", 67
Unité arithmétique et unité centrale, 143, 146, 290
Unité motrice, 262
Unité de traitement spécialisée, 268, 270
Univac 1100/60, 2e p. av. propos, 284
Universel (module E/S), 220, 227, 229, 230
Uranium, 303
Usart, 66

V 24, 234, 238, 240, 241
V 32, 281
Van-Vogt, 18
Varese, 2e p. préface
Variable logique, 18
Variables (noms de), 275
Vax-11/780, 306
Ventilateur, 154
Verrouillage des programmes, 278
Vias, 281
Vidéo, 176
Vie d'un ordinateur, 306
Visualisation sept segments, 64
Visualisation vidéo, 176, 264
Vitesse de transmission, 233, 269
Vitesses de transmission, 233, 269
Vitesses de transmission normalisées, 234
V-MOS, 291, 292
Voice coil, 269
Von Neumann (john), 18, 54
Vortex, 286
VSAM, 270, 271

WD-16, 289
Wafer, 35
Watson (thomas d.), 285
Western digital, 289
Wiley, 320

Wilkes, 57
Write, 277

X-MOS, 291
Xor, 19
XR 210, 195

Yield, 41
Yttrium, 286

Z-80, 289
Z-8000, 299
Zardoz, 286
Zener (diode), 238
Zilog, 289

PLAN DE L'OUVRAGE

1. LES DEUX CULTURES 1
   1-1 Le monde merveilleux du microprocesseur 2
   1-2 D'où venons-nous, où allons-nous ? 5
   1-3 Jetons le pont 7

2. TECHNO-LOGIQUE 9
   2-1 Premier acte : la diode et le transistor 9
      2-1-1 Pourquoi le fer est-il froid ? 9
      2-1-2 Porteur et jonction 10
      2-1-3 La diode à jonction 12
      2-1-4 Le transistor à jonction 14
   2-2 Au mitan des milles portes 17
      2-2-1 Il faut qu'une porte soit ouverte ou fermée 17
      2-2-2 Du comportement de quelques portes fondamentales 18
      2-2-3 Quelques exemples (comparateur de signes, décodeur binaire-décimal) 21
      2-2-4 Les portes négatives de la logique positive : NOR et NAND 23
      2-2-5 Les portes de la création 25
      2-2-6 Un exemple de logique combinatoire : l'autorisation d'accès à la mémoire 27
      2-2-7 Un jeu de bascule électronique : les flip-flops et la logique séquentielle 30
   2-3 Acte deux : les circuits intégrés bipolaires 24
      2-3-1 Un saucisson coûteux 35
      2-3-2 Epitaxie et diffusion 35
      2-3-3 SOS 39
      2-3-4 Masques et artistes 40
      2-3-5 CAO 40
      2-3-6 La course de la puce et de l'ordinateur 41

2-4 Acte trois : le transistor à effet de champ 42
2-5 Dernier acte : les circuits intégrés MOS 43
2-6 Les boîtiers à un franc 45
2-6-1 Le quadruple NAND 45
2-6-2 Les familles (RTL, DTL, TTL, ECL, MOS p et n, C/MOS, Schottky TTL, IZL) 46
2-6-3 Les familles ennemies 47
2-6-4 Sournoises difficultés 48

3. μP : LE MICROPROCESSEUR 49
3-1 Qu'est-ce qu'un microprocesseur 50
3-1-1 Sur une pastille de silicium 50
3-1-2 Le bus : une autoroute à voies multiples parfois encombrées 51
3-2 Brève histoire et tentative impossible (mais pleine d'enseignements) de classement des microprocesseurs 53
3-2-1 Les pionniers, quatre à quatre 53
3-2-2 Architectures Harvard et Princeton 54
3-2-3 Le 8 bits entre en scène 56
3-2-4 Les électrons dépassent les trous, et le 8080 dépasse le 8008 56
3-2-5 Emulation sur les 8 bits canal n 57
3-2-6 Microprogrammes et tranches de machines 57
3-2-7 Tentative de classification 60
3-3 Comment l'esprit déplace la matière 63
3-4 Le Parallel Processing System 4 : PPS-4 65
3-5 Le 8008, pionnir solide au poste 67
3-5-1 Bizarreries 69
3-5-2 Un bon cheval quand même 69
3-6 Les difficultés d'un choix 72
Genèse d'un fils spirituel 73
3.7 Notre mille-pattes de battaille : le 8080 73
3-7-1 Dix fois plus rapide 73
3-7-2 Une nouvelle architecture 74
3-7-3 Bons signaux d'état et de commande : atouts pour le système et le bus 74
3-7-4 Une seconde de soixante kilomètres 76
3-7-5 Mot d'état : une fenêtre sur le CPU 79
3-7-6 Une instruction en pas à pas 79
3-7-7 Format des instructions et des données 81
3-8 Les instructions du 8080 82
3-8-1 Hexa et octal 82
3-8-2 Un peu d'arithmétique 83
3-8-3 Symbolisme des instructions 85
3-8-4 Les instructions de la machine 86
3-8-5 Instructions pour les sous-programmes 102
3-8-6 Exemples de programmation (structurée) 108
3-8-7 Instructions relatives aux interruptions 110
3-8-8 Instructions d'entrée et de sortie 112
3-9 Du côté de la programmation 113
3-9-1 Petite programmation pondérée 113
3-9-2 De la simplicité 117
3-9-3 Doublement à la portée de tous 120
3-9-4 Simplement complexe 121
3-9-5 De la macro à la micro 121
3-9-6 Un dialogue 121
3-9-7 Le B, A, BA des claviers 123
3-9-8 Informaticien et micro-informatique : "Décentralisation/centralisation, ou mieux encore ?" 124

4. DU MICROPROCESSEUR AU MICRO-ORDINATEUR 127
4-1 Per ardua ad astra 127
4-1-1 Conception 127
4-1-2 Les étapes 129
4-2 Statique ou dynamique ? 133
4-3 La série ! 135

5. PROBLEMES PRATIQUES 137
5-1 Manipulation des DIL MOS 138

5-2 Note sur le soudage et les outils 138
5-3 Plan de travail et précautions 139
5-4 Conseils d'assemblage 140
5-5 Description générale de l'unité centrale 143
5-6 Montage de l'unité centrale 146

6. LA MANIPULATION DU PUPITRE 157
6-1 Les modes de fonctionnement 157
6-2 Entrer, relire et exécuter un programme 158
6-3 La roulette électronique 159
6-4 La commande INT 160
6-5 Voyants d'état 160

7. RETOUR AU HARDWARE 161
7-1 Quelques tests en pas à pas 161
7-2 La carte combinée 164
7-2-1 Liaison avec le clavier 164
7-2-2 Fonctionnement de la carte combinée et du bus 165
7-3 Le clavier spécial 168
Programme de scrutation clavier 169
7-4 Alimentations 171
7-4-1 Module 5V 171
7-4-2 Module - 12V 174
7-5 Interface TV 176
7-5-1 Les cartes vidéo 176
7-5-2 Utilisation et fonctionnement de l'interface télévision 176
7-5-3 Accès (timing, contrôle, code) 179
7-5-4 Génération de marges et de caractères, synchro 179
7-5-5 Carte de gestion mémoire 180
7-6 Deuxième fond de panier 181
7-6-1 Solution do-it-yourself 181
7-6-2 Solution de luxe 181
7-7 Interface magnétocassette 181
7-7-1 Description du mode d'enregistrement 181
7-7-2 Le programme de lecture 187
7-7-3 Utilisation des cassettes en Basic 196
7-8 Les microdisquettes 202
7-8-1 Historique du disque souple 202
7-8-2 Le support 204
7-8-3 Formattage et sécurité 206
7-8-4 Organisation de fichiers 206
7-9 Les entrées/sorties 209
7-9-1 Définitions et problèmes généraux 209
Le dilemme 212
Quelques principes 213
Les instructions d'entrée/sortie 214
Structure de l'adresse d'entrée/sortie 214
Les périphériques: notions de moniteur et de système 215
Le bus: signaux, adresse, considérations électriques 215
7-9-2 Entrées Sorties parallèles 220
Module entrée/sortie universel 220
Structure de 8255 220
Modes de fonctionnement 220
Description détaillée (registrs, mot de commande) 222
Utilisation pratique du 8255 227
Module entrée/sortie 8 bits Basic 228
Principe de fonctionnement 228
Caractéristiques électriques 226
Lignes de commande 231
7-9-3 Entrées/Sorties série 231
Structure de l'information 232
Liaison asynchrone 232
Liaison synchrone 234
Le standard CCITT V24 234
Le Module d'entrée/sortie série 238

7-9-4 Carte interface parallèle imprimante 245

8. PROBLEMES ACTUELS DE CONCEPTION 253
8-1 Small is beautiful 253
8-1-1 La technologie hybride: SLT, SLD chez IBM 254
8-1-2 Les circuits intégrés 254
8-1-3 Problèmes de conception des ordinateurs lourds: le mur du son 255
8-1-4 Gros, complexe et simple à la fois 257
8-2 Small is beautiful, again 258
8-2-1 Caveat emptor 260
8-2-2 L'unité motrice 262
8-2-3 Cavalier seul 268
8-2-4 Bis repetita placent 268
8-2-5 L'architecture nodale 271
8-2-6 Le côté Soft 274
8-3 Conception moderne des ordinateurs moyens 280
8-2-1 Les 704 portes d'IBM: à discrétion 281
8-3-2 Une bibliothèque métallique 282
8-3-3 Conception structurée pour tests automatiques 283
8-3-4 Le grand pavois 284
8-3-5 L'Univac 1100/60: un gros ordinateur entièrement réalisé avec des microprocesseurs 284
8-4 Le superordinateur supraconducteur 285
8-4-1 La Fonction Josephson mise en pratique 285
8-4-2 De Josephson à Zardoz 286

9. LES NOUVEAUX MICROS 287
9-1 Les destins des micros: un diorama en 16 287
Le PACE 287
Le MicroNOVA nN 601 288
Les TI 9900 288
Le MCP-1600: PASCAL Microengine 289
9-2 Les nouveaux 8 bits 289
Tout sur la puce: 8021/48/39, 3870 290
Les bizarres 290
9-3 Les nouvelles technologies 291
Le H-MOS 291
Le V-MOS (et le D-MOS japonais) 291
L'I3L et les Fairchild 9440/45 292
9-4 Le premier des nouveaux 16 bits: le 8086 293
9-4-1 Technologie de 8086 293
9-4-2 Extension de l'architecture 295
9-4-3 La queue 295
9-4-4 L'adressage 296
9-4-5 Le jeu d'instructions et les modes d'adressage 297
9-4-6 Interruptions, multitraitement 298
9-4-7 Le pont micro-mini 299
9-5 Les 16 bits qui arrivent 299
9-5-1 Le Z 8000: adressage, types de données, registres, les 8 modes d'adressage, la vitesse 299
9-5-2 Le Motorola MC 68000: vers les 32 bits 300
9-6 Les 1 bit: 301
9-7 Les plus-que-8-bits 301
9-7-1 Le 6809: réutilisation des ROM, statistiques et nouvelles instructions, le super 8 bits 301
9-7-2 Le 8088: le pintade est un animal élégant! Une démarche inverse du 6809, la queue plus courte 302
9-8 L'horizon prévisible 303
9-8-1 Mémoires et particules alpha: un défi difficile 303
9-8-2 Considérations de conception pour le proche avenir 303
9-8-3 Les sources d'erreurs dans un ordinateur 305
9-8-4 La limite ultime 309

ANNEXE: Ecrivez votre propre assembleur pour le 8080 311

Composition par Madison, 147, rue de Courcelles, 75017 Paris - Imprimé par J.M. Barbou - 29, rue A.-Joineau - 93310 Le Pré-Saint-Gervais

Cet ouvrage tient une place unique dans la littérature mondiale consacrée aux microprocesseurs.

Ses auteurs, lorsqu'ils l'ont entrepris, avaient déjà réalisé le best-seller de l'initiation à l'informatique. (Un Fil d'Ariane tome 1 : l'ordinateur à la portée de tous) qui en est à sa quatrième édition et sixième réimpression (adopté par Peugeot, l'EDF, la Sécurité Sociale, et lu par une centaine de milliers d'utilisateurs... et d'informaticiens) (1).

Ce nouveau volume de la collection "Un Fil d'Ariane", utilisant la même approche originale, vous mène aussi pas à pas à une découverte, celle du microprocesseur sous tous ses aspects.

De manière réellement compréhensible pour un lecteur isolé, alliée à une surprenante richesse d'informations, il part d'un niveau zéro pour aboutir très progressivement à une connaissance palpable, physique, concrète du fonctionnement interne, de l'utilisation et de la programmation des microprocesseurs.

En fait, les auteurs retracent en détail leur propre expérience : comment ils ont entièrement conçu et réalisé Alcyane, un micro-ordinateur d'usage général, gestion et scientifique (ils ont à cette époque fondé une société, MBC, aujourd'hui l'un des rares constructeurs français d'ordinateurs).

Les plus récents développement sont couverts : 8 bits modernisés, mixtes 8/16 bits, divers types de 16 bits, 16/32 bits, sans oublier les nouveaux 1 bit.

C'est l'alliance constante du théorique, de la pédagogie et du concret, accompagnée d'un humour toujours sous-jacent dans le texte et les illustrations qui rend cet ouvrage, si profondément original, utile à un très large public.

Il rendra des services fondamentaux aux catégories suivantes de lecteurs (entre autres...) :
— les informaticiens "classiques" qui ne veulent pas se laisser dépasser, dans les entreprises ou les SSCI,
— les responsables, ingénieurs, techniciens de PMI ou grandes sociétés qui veulent incorporer des microprocesseurs dans leurs produits pour les rendre "intelligents",
— les électroniciens "classiques" qui ne veulent pas se laisser dépasser,
— les utilisateurs actuels ou potentiels de micro-ordinateurs, en gestion, en scientifique ou en industriel,
— les étudiants de lycées techniques, IUT, UER, etc.,
— les professeurs, animateurs, intervenants, responsables de séminaires,
— tous ceux enfin qui désirent s'initier aux microprocesseurs, seuls ou dans le cadre d'une formation continue.

---

(1) Il leur a valu la médaille du SPREF (Syndicat de la presse radio-électronique française) pour la qualité de l'œuvre de vulgarisation informatique.

LES AUTEURS

Jean-Pierre BOUHOT
Ingénieur, ancien de l'aviation, passé par General Electric, professeur honoraire d'informatique à l'Ecole Centrale, rédacteur en chef de l'Informatique Nouvelle. Grand amateur de Science-fiction. Marin à ses heures. Co-fondateur de MBC.

Georges COTTIN
Ingénieur, ancien de la marine, passé par Hewlett-Packard. Omniscient en conception à base de microprocesseurs. Grand amateur de Science-fiction. Aviateur à ses heures. Co-fondateur de MBC.

Jean TRICOT
Ingénieur, ancien du conseil, passé par les bases de données de la SNCF. Grand amateur de Science-fiction. Joueur d'échecs à ses heures, auteur d'ouvrages et de chroniques remarqués en informatique.

ISBN 2-901001-09-2